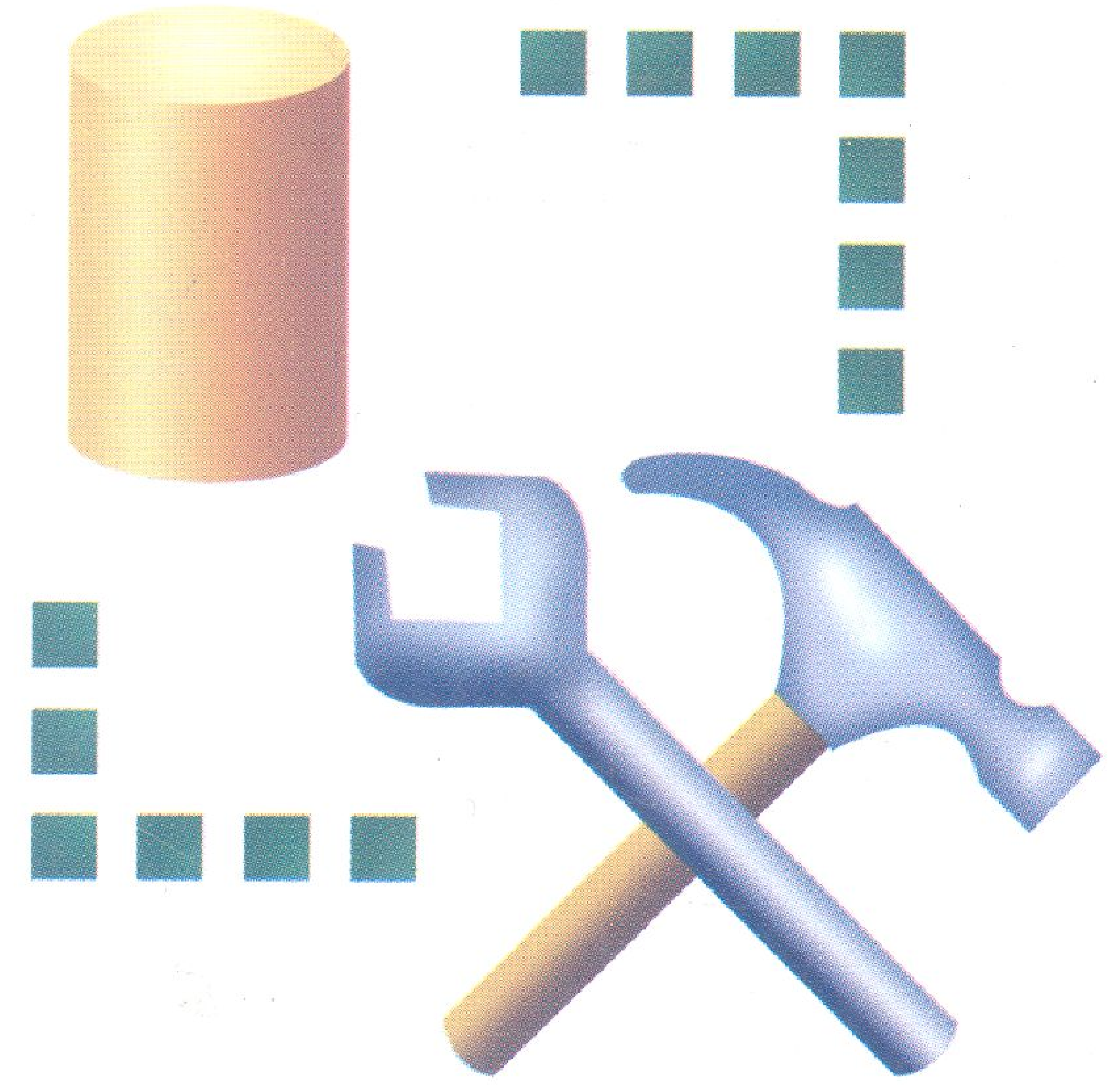
من الصفر
إلى الاحتراف

# برمجة قواعد البيانات

## فى فيجيوال بيزيك دوت نت ٢٠١٠



مهندس
**محمد حمدي غانم**

**Database Programming**
**In VB.NET 2010**

دار المعرفة

من الصفر إلى الاحتراف:

# برمجة قواعد البيانات

## في فيجيوال بيزيك دوت نت ٢٠١٠

بقلم:

م. محمد حمدي غانم

الناشر:

مكتبة دار المعرفة

٢٠١٠

- اسم الكتاب : برمجة قواعد البيانات فى فيجوال بيزيك دوت نت 2010
- اسم المؤلف : م / محمد حمدى غانم
- الناشـــــــر : مكتبة دار المعرفة
- رقم الإيداع بدار الكتب : 5640 /2010
- الترقيم الدولي : 0 – 88 - 5423 - 977

(c) حقوق النشر والطبع والتوزيع محفوظة لمكتبة دار المعرفة
لا يجوز نشر جزء من الكتاب أو إعادة طبعه أو اختصاره بقصد الطباعة أو اختزان مادته العلمية أو نقله بأي طريقة سواء كانت إلكترونية أو ميكانيكية أو بالتصوير أو خلاف ذلك دون موافقة خطيه من الناشر مقدما

مكتبة دار المعرفة
8 شارع يوسف الزينى أمام هندسة عين شمس ت:24850310
4 شارع السرايات أمام هندسة عين شمس ت:26844043
E-mail:dar_elmaarefa@yahoo.com

# مقدمة الناشر

الحمد لله رب العالمين والصلاة والسلام على أشرف المرسلين سيدنا محمد صلى الله عليه وسلم وبعد.....

إن أول ما أنزل من القرآن على رسولنا الحبيب قوله سبحانه وتعالى :

اقْرَأْ بِاسْمِ رَبِّكَ الَّذِي خَلَقَ ﴿١﴾ خَلَقَ الْإِنسَانَ مِنْ عَلَقٍ ﴿٢﴾ اقْرَأْ وَرَبُّكَ
الْأَكْرَمُ ﴿٣﴾ الَّذِي عَلَّمَ بِالْقَلَمِ ﴿٤﴾ عَلَّمَ الْإِنسَانَ مَا لَمْ يَعْلَمْ ﴿٥﴾

كانت هذه الآيات العظيمة أول ما أنزل من الوحى على رسولنا الكريم صلى الله عليه وسلم دعوة من الله سبحانه للقراءة والعلم، وعندما فهم المسلمون الأوائل هذا الدين الكريم وعملوا بأوامره، حملوا مشاعل العلم الى العالم بأسره، وأخرجوا العالم من ظلمات الجهل والكفر إلى نور العلم والإيمان.. وعندما ابتعدنا عن ديننا القويم انتقلت رايات العلم من أيدينا إلى أيادىَ أخرى فتقدمت بلادهم بالعلم وتأخرت بلادنا.

وإيمانا بهذا النهج القويم الذي دعا إليه ديننا العظيم من الحث على القراءة والتعلم، تسعى مكتبة دار المعرفة إلى نشر كل ما هو جديد وحديث فى مجالات المعرفة، خاصة تقنية الحاسب والعلوم الهندسية، معتمدة فى ذلك على ذوى الخبرة والكفاءة لتُقدّم للقارئ العربى فى كل مكان بأسلوب مبسط وميسر، يعتمد على التسلسل المنطقى فى العرض، ليخدم بذلك القارئ المبتدئ والمحترف، مزودا بالأمثلة التوضيحية والتطبيقات العملية، لإيضاح الفكرة وترسيخ المعلومة لدى القارئ.. ولقد ألحقنا قرصا مرنا ببعض الكتب تيسيرا للقارئ لدى تعامله مع الحاسب الألى.

كما تقوم مكتبة دار المعرفة باستراد أحدث الكتب والمراجع العلمية، لمواكبة كل ما هو جديد فى مجالات العلوم، لتوفيره للباحثين والطلاب بأسعار معتدلة.. وزودت المكتبة بقاعة للاطلاع عسى أن يلقى هذا العمل القبول لدى الله سبحانه وتعالى ويجعله من العلم النافع الذى ينتفع به.

ويسر المكتبة تلقى مقترحاتكم وتعليقاتكم لتكون موضع الدراسة والاهتمام.

**الناشر**

## مقدمة الكاتب

**بسم الله، والحمد لله، والصلاة والسلام على رسول الله، وبعد:**

تعتبر قواعد البيانات القلب المركزي في معظم أنواع المشاريع، سواء كانت تعمل على سطح المكتب Desktop أو موقع إنترنت Web Site أو شبكة Network أو جهاز يدوي محمول.. لهذا يعتبر تعلم كيفية إنشاء قواعد البيانات والتعامل معها أمرا لا غنى عنه لتنفيذ المشاريع التجارية والإدارية والعملية، التي تتيح للمستخدم استرجاع البيانات وتعديلها وحفظها والبحث فيها بمختلف الطرق.

ويبدأ هذا الكتاب الرحلة معك من الصفر، ليعرفك على المفاهيم الأساسية اللازمة للتعامل مع قواعد البيانات، وكيف تنشئها في Access و SQL Server 2008، وكيف تكتب الاستعلامات التي تحصل على البيانات منها باستخدام لغة SQL.

بعد هذا يعلمك الكتاب كيف تتعامل مع قواعد البيانات من داخل مشاريع فيجيوال بيزيك دوت نت باستخدام تقنية ADO.NET، لتستطيع الاتصال بقاعدة البيانات، وطلب السجلات منها، وكيف تقوم بحفظها مرة أخرى في قاعدة البيانات إذا دخلت عليها أية تعديلات.

ويعلمك الكتاب أيضا كيف تعرض البيانات للمستخدم من خلال تقنية الربط Binding، ويشرح بالتفصيل أهم الأدوات المخصصة لهذا الغرض، مثل موجه الربط BindingNavigator ومصدر الربط BindingSource وجدول العرض DataGridView ومكرر البيانات DataRepater.

***

ويشرح الكتاب بالتفصيل أكثر من ٥٠ مشروعا متنوعا تغطي محتوياته، لتتعلم من خلالها:

- كيف تحصل على البيانات من قواعد البيانات بمختلف الطرق، سواء باستخدام قارئ البيانات DataReader أو موصل البيانات DataAdapter أو موصل الجدول TableAdapter.
- كيف تحتفظ بالبيانات في الذاكرة، باستخدام مجموعة بيانات DataSet سواء كانت عادية أو محددة النوع Typed.
- كيف تنقل البيانات بين نوعين مختلفين من قواعد البيانات.
- كيف تحفظ البيانات الثنائية Binary Data في ملفات مستقلة على الخادم خارج قاعدة البيانات في SQL Server 2008.

- كيف تعرّف المعاملات Parameters والمعاملات الجدولية Table-Valued Parameters، وكيف تستخدمها لتمرير البيانات إلى الإجراءات المخزنة في Sql Server 2008.
- كيف تحمي قاعدة البيانات من القراصنة الذين يحاولون دس الاستعلامات SQL Injection من خلال مشروعك.
- كيف تقرأ البيانات الثنائية والنصية الضخمة تتابعيا Sequentially على صورة أجزاء في SQL Server 2008.
- كيف تنشئ الإجراءات المخزنة في Access.
- كيف تحفظ البيانات في ملف XML وكيف تستعيدها منه مرة أخرى.
- كيف تستخدم مخطط XML لإنشاء مجموعات بيانات خاصّة Custom DataSet لا تعتمد على قاعدة بيانات.
- كيف تتعامل مع علاقة واحد بمتعدد One-To-Many Relation، وعلاقة متعدد بمتعدد Many-To-Many Relation، والعلاقة الذاتية Self Relation.
- كيف تستخدم مصانع المزودات Provider Factories لكتابة فئات عامة قادرة على التعامل مع أي نوع من قواعد البيانات، مما يختصر الكود الذي تكتبه، و يمهد لك الطريق لإنشاء مشاريع متعددة الطبقات N-Tier Applications.
- كيف تحل مشاكل تصارع أكثر من مستخدم على حفظ البيانات في نفس اللحظة باستخدام التطابق المتفائل Optimistic Concurrency.
- كيف تعرض البيانات في اللافتات ومربعات النص والقوائم والجداول، وكيف تربط كل هذه العناصر معا.
- كيف تنشئ أنواعا جديدة من أعمدة جدول العرض، تعرض خاناتها أداة اختيار التاريخ أو شجرة منسدلة أو أي نوع آخر تريده من الأدوات.
- كيف تجعل جدول العرض يعمل في الوضع الافتراضي Virtual Mode وكيف تضيف إليه تقنية تقسيم السجلات على صفحات Paging.
- كيف تنشئ قالبا لعرض كل سجل، وكيف تكرر عرضه باستخدام مكرر البيانات DataRepeter.
- كيف تستخدم مكرر البيانات في الوضع الافتراضي Virtual Mode.

وغير هذا الكثير.

***

ويغطي هذا الكتاب بالتفصيل حوالي ١٣٥ واجهة وفئة وسجلا من مكتبة إطار العمل، مخصصة للتعامل مع تطبيقات قواعد البيانات، شارحا خصائص ووسائل وأحداث هذه المكونات بالتفصيل.. لهذا يعتبر الكتاب مرجعا مفصلا مبوبا، يمكن لقارئه الرجوع إليه عند البحث عن تفاصيل أي فئة أو خاصية أو وسيلة أو حدث، في نفس الوقت الذي يجعله صالحا للقراءة ككتاب تعليمي عملي مرتب من الأسهل إلى الأصعب، ينقل إلى المبرمج في صفحات معدودات خبرة سنوات في برمجة تطبيقات قواعد البيانات، ويرشده إلى كيفية حل المشكلات غير المتوقعة التي تواجهه في هذا المجال، وكيف يحسن أداء برنامجه بتوفير أكبر قدر من الذاكرة، وكيف يحافظ على كفاءة خادم البيانات، بتقليل عدد الاتصالات ووقت كل اتصال بقدر الإمكان.

باختصار: هذا هو الكتاب الذي تبحث عنه.

**والله ولي التوفيق**

## لمن هذا الكتاب:

رغم أن هذا الكتاب يفترض أن قارئه لا يمتلك أية معرفة مسبقة بقواعد البيانات والبرامج التي ينشئها بها، فإنه على الجانب الآخر، يشترط في قارئه أن يكون على دراية بلغة فيجيوال بيزيك دوت نت، وأن يجيد المتطلبات التالية:

- أساسيات كتابة الكود بلغة فيجيوال بيزيك دوت نت، كتعريف المتغيرات وكتابة جمل الشرط وحلقات التكرار Loops، وكتابة واستدعاء الدوال Functions.
- أساسيات ومفاهيم البرمجة الموجهة بالكائنات OOP، كالفئات Classes والواجهات Interfaces والوراثة Inheritance.
- أساسيات التعامل مع إطار العمل، وفئاته الرئيسية، خاصة المجموعات Collections والملفات Files وفئات معلومات الثقافة CultureInfo.
- أساسيات التعامل مع مشاريع الويندوز، والأدوات المختلفة كمربع النص TextBox ومربع الاختيار CheckBox والقوائم Lists.

فإذا لم تكن تجيد هذه الأساسيات، فننصح بقراءة كتبنا السابقة:

١- "من الصفر إلى الاحتراف: فيجيوال بيزيك دوت نت".
٢- "من الصفر إلى الاحتراف: برمجة إطار العمل".
٣- "من الصفر إلى الاحتراف: برمجة تطبيقات الويندوز".

## الرموز المستخدمة في هذا الكتاب:

| | |
|---|---|
| | سجل Structure. |
| | فئة Class. |
| | واجهة Interface. |
| | ثابت Constant. |
| | خاصية Property يمكنك قراءة أو تغيير قيمتها. |
| | خاصية للقراءة فقط Read Only Property. |
| | وسيلة Method. |
| | معامل Operator. |
| | حدث Event. |
| | هذا العنصر مشترك Shared، يمكن استخدامه عبر اسم الفئة مباشرة. |

# تركيب قواعد البيانات

# Database Structure

سنتعرف في هذا الفصل على أهم التعريفات والمفاهيم الأساسية التي تلزمنا للتعامل مع قواعد البيانات في باقي فصول هذا الكتاب.

## قاعدة البيانات:

قاعدة البيانات Database هي ملف أو مجموعة من الملفات، تستخدم لتخزين البياناتِ التي تربطها علاقات معيّنة، مثل الجداول.. وتتميز قاعدة البيانات عن الملفات العادية، بوجود برامج معينة للتعامل معها، مما يجعل حفظ البيانات وقراءتها والبحث فيها أمورا أكثر سهولة وتنظيمًا، وخالية من الأخطاء.. هذه البرامج تسمّى "أنظمة إدارة قواعد البيانات":
Database Management Systems (DBMS)
وهناك أنواع كثيرة من قواعد البيانات، تبعا للشركة التي تنتجها والبرامج التي تنشئها، وطريقة تنسيق البيانات المحفوظة في ملف قاعدة البيانات.. فهناك ــ على سبيل المثال ــ قواعد بيانات Access وقواعد بيانات SQL Server وهما من إنتاج ميكروسوفت، وهناك قواعد بيانات Oracle و MySQL، وغيرها من برامج قواعد البيانات من إنتاج شركات أخرى.
ومهما كانَ نوع قاعدة البيانات التي تتعامل معها، ومهما كانت طريقة تخزينها في الملفّ، فإنّ كل قواعد البيانات تتبع معايير أساسيّة وتحقق شروطا معيّنة متعارفا عليها دوليّا، كما أنّها كلها بلا استثناء تستخدم "لغة الاستعلام المركّبة" Structured Query Language (SQL)، وهي لغة خاصّة تستخدم لحفظ واسترجاع وتحديث البيانات الموجودة في قواعد البيانات.
وتحتوي مجموعة البيانات على أنواع مختلفة من العناصر، منها:

- الجداول Tabled.
- العلاقات Relations.
- القيود Constrains.
- العروض Views.

## الجداول Tables:

تتركب قاعدة البيانات في الأساس من مجموعةٍ من الجداول، هي التي يتم تخزين البيانات بها.. فإذا لم تحتوِ قاعدة البيانات على أي جداول، كانت فارغة بلا قيمة. فمثلا، لو لديك قاعدة بيانات خاصة بمكتبة، فسيوجد بها جدول يحتوى على عناوين الكتب الموجودة في المكتبة (وليكن اسمه Books)، وجدول آخر يحتوى على أسماءِ مؤلفي هذه الكتب (وليكن اسمه Authors)، وجدول ثالث يحتوى على أسماء دور النشر التي نشرت هذه الكتب (وليكن اسمه Publishers).
ويتكون كل جدول من:

**أ. أعمدة Columns (وتسمّى حقولا Fields أيضا):**

ويشبه العمود المتغيرات في لغات البرمجة، فله اسم ونوع.. وتختلف أسماء أنواع البيانات اختلافا طفيفا من قاعدة بيانات إلى أخرى، لكنها في النهاية تشبه أنواع البيانات الموجودة في لغات البرمجة، مثل الأعداد الصحيحة Integers، والأعداد الطويلة Long وغيرها.

وعند إنشاء الجدول، يجب تصميم الأعمدة بشكل صحيح لمراعاة التالي:

١- أن يكون اسم العمود معبرا عن البيانات التي سيتم تخزينها فيه.

٢- أن يتم اختيار نوع البيانات المناسب لحفظ بيانات هذا العمود، بحيث:

- لا يكون كبيرا جدا فيهدر مساحة كبيرة على القرص الصلب ويبطئ من عمليات نقل البيانات عبر الشبكة.
- ولا يكون صغيرا جدا فيعجز عن استيعاب بعض البيانات الهامة، مما يسبب أخطاء في البرامج التي تحاول الكتابة في الجدول، أو يمنعها من كتابة بعض البيانات الهامة إن كانت هذه البرامج مصممة جيدا لمعالجة الأخطاء التي تحدث.

**ب. صفوف Rows (وتسمّى سجلات Records أيضا):**

يحتوي الجدول على صفوف، كل صف منها يعتبر وحدة بيانات مستقلة، تحفظ القيم فيها تحت كل عمود من أعمدة الجدول.. فمثلا: لو لديك جدول خاص بتلاميذ إحدى المدارس، فسيحتوي هذا الجدول على أعمدة تمثل رقم الطالب واسمه وعمره وفصله، وسيكون لكل تلميذ صف (سجل) في هذا الجدول يحتوي على المعلومات الخاصة به، كما هو موضح:

| Class | Age | Name | ID |
|---|---|---|---|
| ٦ | ١٢ | أحمد علي | ١ |
| ٥ | ١١ | محمود رشوان | ٢ |
| ١ | ٦ | شريف عمران | ٣ |

ويمكن أن يحتوي الجدول على مئات أو آلاف الصفوف تبعا لاحتياجك.. وتتحكم العوامل التالية في الحد الأقصى من الصفوف المسموح بحفظه في جداول قاعدة البيانات:

١- مساحة التخزين المتوفرة على القرص الصلب الذي توجد عليه قاعدة البيانات.. وإن كانت بعض أنواع قواعد البيانات تسمح بتقسيم ملفاتها على أكثر من قرص صلب، للسماح بحفظ ملايين السجلات.

٢- نوع قاعدة البيانات.. حيث إن بعض أنواع قواعد البيانات مثل Access مصممة للتعامل مع قواعد بيانات صغيرة نسبيا، لهذا ينصح بألا تزيد عدد السجلات المحفوظة في قاعدة البيانات الخاصة بها عن ١٠٠ ألف سجل، أو مليون سجل على أقصى تقدير!.. أعرف أن هذا يبدو رقما ضخما بالنسبة لك، ولكن في قواعد البيانات العملاقة هذا رقم في غاية التفاهة، وقواعد بيانات SQL Server و Oracle تتعامل مع سجلات تتجاوز عشرات وربما مئات الملايين!

٣- نوع العمليات التي تجريها على الجدول.. فلو كنت تحتاج إلى إجراء عمليات كثيرة على الجداول، وكان هناك عدد كبير من المستخدمين الذين يتعاملون مع هذه الجداول في نفس اللحظة، فإن ضخامة أحجام الجداول تؤدي إلى إبطاء نقل البيانات وإبطاء عمليات الإضافة والحذف والبحث.. في مثل هذه الحالة ينصح بتقسيم الجدول إلى جداول أصغر فيما يعرف باسم عملية التطبيع Normalization.. هذه العملية يمكن أن تتم:

أ. رأسيا: وذلك بتقسيم الأعمدة على أكثر من جدول، مع ربطها معها باستخدام رقم السجل ID والذي يعرف في هذه الحالة باسم المفتاح Key.

ب. أفقيا: وذلك بتقسيم الصفوف على أكثر من جدول تبعا لعامل مشترك بينها.. مثلا: يمكن تقسيم الصفوف على حسب تواريخها، وبدلا من أن يكون لديك جدول واحد لكل السنوات، يكون لديك جدول لكل سنة (هذا مناسب لجدول التلاميذ على سبيل المثال).. كما يمكنك أن تقوم بعملية تطبيع لقاعدة البيانات كلها، بتقسيمها إلى عدد من قواعد البيانات، تبعا للعام، أو المنطقة الجغرافية إلخ.

ويتحكم نوع المشروع الذي تتعامل معه وطبيعة بياناته، في كيفية حل مشاكل تضخم قواعد البيانات، وكيفية تقسيمها.. لهذا من المهم أن تحلل النظام الذي تتعامل معه جيدا لاختيار أكفأ تصميم لقاعدة البيانات منذ البداية، لأن محاولة حل مشكلة ظهرت بعد عدة سنوات من العمل على النظام، قد تكون مكلفة جدا ومضيعة للوقت والجهد، وربما تؤدي إلى إعادة كتابة البرامج كلها منذ البداية!

## قواعد البيانات المترابطة Relational Databases:

لو كانت لدينا قاعدة بيانات للمكتبات ودور النشر، فسيحتوي جدول الكتب Books على عدّة أعمدة لتخزين بعض المعلومات عن كل كتاب، مثل اسمه، وعدد صفحاته، وموضوعه وسعره... إلخ.

يجب أيضا أن يحتوي جدول الكتب على معلومات عن مؤلف كل كتاب، مثل اسمه ونبذة عنه وبريده الإلكتروني وهاتفه، ليستفيد بها الناشر عندما يريد الاتصال به.. فكيف يا ترى نحفظ هذه المعلومات؟

بداهة، ليس من الذكي وضع كل هذه المعلومات في جدول الكتب، لأن سيحتوى بالتأكيد على أكثر من كتابٍ لنفس المؤلف، مما سيسبب المشاكل التالية:

- تكرار بيانات المؤلف أكثر من مرّة، مما يزيد من حجم قاعدة البيانات ويبطئ عمليّات البحث.
- إرهاق المستخدم الذي يدخل البيانات، بسبب تكرار كتابة بيانات نفس المؤلفين مرارا وتكرارا.
- احتمال حدوث خطأ في إدخال بيانات المؤلف المكررة، مما يؤدي لتضارب في بيانات نفس المؤلف.
- عند تغير رقم هاتف المؤلف، يجب تعديل هذا في كل سجلات الكتب!
- عند البحث عن بيانات المؤلف، سنحصل على أكثر من سجلات بها نفس التفاصيل بلا فائدة!

والأمر نفسه بالنسبة لبيانات دار النشر التي نشرت الكتاب، فمن المتوقع وجود مئات الكتب لكل دار نشر، ومن العبث تكرار معلومات مثل اسم وعنوان وهاتف دار النشر، واسم الناشر أو مدير الدار... إلخ مع كل كتاب!

إذن فما هو الحل المناسب لهذه المشكلة؟

كما ذكرنا من قبل، يجب إجراء عملية تطبيع Normalization لهذه البيانات، وذلك بتخصيص جدولٍ للمؤلفين Authors يحتوى على سجل واحد لكل مؤلف يحتوي على بياناته الأساسية، وجدول للناشرين Publeshers يحتوي على سجل

واحد فقط لكل دار نشر به تفاصيلها، مع ربط جدول الكتب بكل من هذين الجدولين.. دعنا نرى كيف نفعل هذا:

في جدول المؤلفين سننشئ عمودا للرقم المسلسل لكل مؤلف، اسمه ID.. وفي جدول الكتب سننشئ عمودا اسمه AuthorID، لنضع فيه رقم مؤلف كل كتاب.. فمثلا لو كان رقم (توفيق الحكيم) في جدول المؤلفين هو ١، فإنّ الخانة AuthorID في جدول الكتب لا بدّ أن تكون ١ لكل من الكتب التالية:

"شهرزاد" و"الأيدي الناعمة" و"أرني الله" و"شجرة الحكم"

كما هو موضح في الجدولين التاليين:

| About | eMail | Phone | Author | ID |
|---|---|---|---|---|
| – | | | توفيق الحكيم | ١ |
| – | | | عباس العقاد | ٢ |
| – | | | محمد عبد الحليم عبد الله | ٣ |
| – | | | نبيل فاروق | ٤ |
| – | aktowfik@hotmail.com | | أحمد خالد توفيق | ٥ |
| – | msvbnet@hotmail.com | | محمد حمدي | ٦ |

| Price | Pages | AuthorID | Book | ID |
|---|---|---|---|---|
| ٥٠٠ | ٢٠٠ | ١ | أرني الله | ١ |
| ٧٠٠ | ٣١٥ | ١ | يوميّات نائب في الأرياف | ٢ |
| ٣٥٠ | ١٥٠ | ١ | عصا الحكيم | ٣ |
| ٥٠٠ | ٢٧٠ | ٢ | سارة | ٤ |
| ٦٠٠ | ٣٢٩ | ٢ | عبقرية محمد | ٥ |
| ٦٠٠ | ٢٨٠ | ٢ | عبقرية عمر | ٦ |
| ٤٠٠ | ٢٣٠ | ٣ | شجرة اللبلاب | ٧ |
| ٢٠٠ | ١٠٠ | ٤ | مهنتي القتل | ٨ |
| ٢٠٠ | ١١٠ | ٤ | الأفق الأخضر | ٩ |
| ٢٠٠ | ١٣٠ | ٥ | أسطورة الغرباء | ١٠ |
| ٢٥٠ | ١٧٨ | ٥ | لا تدخلوا شيرود | ١٢ |
| ١٠٠ | ٦٠ | ٦ | مجرد طريقة للتفكير | ١٣ |
| ٢٠٠ | ١٨٠ | ٦ | حائرة في الحب | ١٤ |
| ١٠٠ | ٦٠ | ٦ | بين قوسين من الخلود | ١٥ |

نفس الشيء يمكن فعله بالنسبة لجدول الناشرين، حيث سنضع فيه تفاصيل كل الناشرين، ونربط بينه وبين جدول الكتب بإنشاء عمود اسمه PublisherID في جدول الكتب.

والآن، دعنا نؤكد مرة أخرى فوائد هذه العملية:

١- توفير مساحة التخزين، فبدلا من تكرار نفس بيانات المؤلفين والناشرين مع كل كتاب، صرنا نكرر فقط رقم المؤلف ورقم الناشر.

٢- تسريع عمليّة البحث.. فمثلا لو أردت البحث عن كل كتب توفيق الحكيم في جدول الكتب، فسيكون البحث عن الرقم ١ في الحقل AuthorID (وهو حقل عددي) أسرع بكثير من البحث في الحقل Author (وهو حقل نصي) عن الاسم "توفيق الحكيم".. صحيح أننا سنبحث في جدول المؤلفين أولا عن "توفيق الحكيم" للحصول على الرقم الخاص به، لكن عدد الصفوف في جدول المؤلفين سيكون أقل بكثير من عدد الصفوف في جدول الكتب.

٣- سهولة التعديل وتصحيح الأخطاء.. افترض مثلا أنّ الموظف الذي يُدخل أسماء المؤلفين أخطأ وكتب الاسم "توفيق الحكيم" كالتالي: "تفيق الحكيم".. في هذه الحالة ما عليه إلا أن يعدّل الخطأ مرّة واحدة في جدول المؤلفين.. أمّا لو كانت أسماء المؤلفين موجودة في نفس جدول الكتب، فإنّ تكرار كتابة اسم المؤلف يجعل احتمالات الخطإ أكبر، بالإضافة إلى تضييع الوقت والجهد في كتابتها، ومشقة تعديلها كلها!

وعند استخدام هذا التنظيم، يقال إن هناك علاقة Relationship بين جدولي الكتب والمؤلفين، حيث إنّ الرابط بينهما هو رقم المؤلف (Author.ID و Books.AuthorID).. كما توجد علاقة بين جدول الناشرين وجدول الكتب، والرابط بينهما هو رقم الناشر (Publisher.ID و Books.PublisherID).

هنا يجب أن نتعرف على المصطلحات التالية:

- يسمّى الحقل المشترك في كلا الجدولين "**حقلا مفتاحيا**" **Key field**.
- يسمّى الحقل ID في جدول المؤلفين باسم "**المفتاح الأساسيّ**" **Primary Key**، وذلك لأنه متفرّد Unique غير قابل للتكرار (يجب ألا يتكرر نفس المفتاح لأكثر من مؤلف).. ولا يحتوي الجدول على أكثر من مفتاح أساسيّ واحد.
- يسمّى الحقل AuthorID في جدول الكتب باسم "**المفتاح الفرعيّ**" أو "**المفتاح الدخيل**" **Foreign Key**، لأنّه موضوع في هذا الجدول ليربطه بجدول آخر.. ويمكن أن تتكرّر بعض القيم في خانات هذا العمود بدون أدنى مشكلة (كما في حالة وجود أكثر من كتاب لنفس المؤلف).. ويمكن

أن يحتوي الجدول على أكثر من مفتاح فرعيّ تربطه بعدد من الجداول الأخرى (كأن يحتوي جدول الكتب على المفتاحين الفرعيين AuthorID و PublisherID).

- يسمّى جدول المؤلفين باسم "**الجدول الرئيسيّ**" **Master**، وجدول أسماء الكتب باسم "**جدول التفاصيل**" **Details**، حيث يمكنك أن تحصل منه على معلومات عن الكتب المتوافرة لكل مؤلف.. وتسمى العلاقة بين الجدولين بعلاقة **Master-Details**.
- تسمّى عمليّة مطابقة الصفوف بينَ الجدولين المترابطين باستخدام الحقل المفتاحيّ، بعملية "**الربط**" **Join**.. كأن تحاول معرفة اسم مؤلف كتاب معيّن، أو أن تحاول أن تعرف كل الكتب التي كتبها نفس المؤلف.

## أنواع العلاقات:

يوجد أربعة أنواع رئيسية للعلاقات بين الجداول:

١- **علاقة "واحد بمتعدّد" One-to-many Relationship:**

تسمّى العَلاقة بين جدول المؤلفين وجدول الكتب في المثال السابق بعلاقة "واحد بمتعدّد" One-to-many Relationship، وذلك لأنّ مؤلفًا واحدا يمكن أن يكون له أكثر من كتاب.

٢- **علاقة "متعدّد بمتعدّد" Many-to-many Relationship:**

افترض أن لدينا كتابا اشترك في تأليفه أكثر من مؤلف.. في هذه الحالة لن نضع العمود AuthorID في جدول الكتب، وبدلا من هذا سننشئ جدولا ثالثا هو جدول "مؤلفو الكتب" BooksAuthors.. هذا الجدول يسمى **جدول الربط Link Table** أو **جدول الاتصال Junction table**، وهو يتكون من عمودين فقط: رقم الكتاب BookID ورقم المؤلف AuthorID كالتالي:

| AuthorID | BookID |
|---|---|
| ٢٣ | ١ |
| ١٠٦ | ١ |
| ٨ | ٢ |
| ٨ | ٣ |

حيث اشترك المؤلفان رقما ٢٣ و ١٠٦ في تأليف الكتاب رقم ٢، بينما قام المؤلف رقم ٨ بتأليف الكتابين رقمي ٢ و ٣.. هذه هي علاقة متعدّد بمتعدّد Many-to-many.

٣- علاقة "واحد إلى واحد" **One-to-one:**

في هذه العلاقة يتم تقسيم السجل الواحد على جدولين، وذلك إذا كان عدد أعمدة الجدول كبيرا جدا، وكان بعضها قليل الأهمية ونادر الاستخدام، أو إذا كانت هذه الأعمدة تحتوي على بيانات سرية، لهذا يجب وضعها في جدول خاص ومنح صلاحية التعامل معها لمدير النظام مثلا.. بهذه الطريقة يكون جزء من السجل في الجدول الأول متاحا لكل المستخدمين، وباقي السجل في جدول آخر مخصص لمستخدمين بعينهم.

وفي علاقة "واحد بواحد" يكون كل من المفتاح الأساسي والمفتاح الفرعي متفردا Unique غير قابل للتكرار، وبهذا نضمن ارتباط كل مفتاح أساسي بمفتاح فرعي واحد فقط.

وكمثال: افترض أن لدينا جدولا لبيانات الموظفين اسمه Employees يبدو كالتالي:

| ID | Employee | Phone | eMail | Address |
|---|---|---|---|---|
| ١ | أحمد شريف | .... | .... | .... |
| ٢ | مجدي نوارة | .... | .... | .... |
| ٣ | صالح رشدان | .... | .... | .... |
| ٤ | ياسر الهواري | .... | .... | .... |

في هذه الحالة سنحتاج إلى جدول آخر نضع فيه اسم المستخدم وكلمة السر الخاصة بكل موظف، وليكن اسمه EmployeesPasswords، وسنربطه بالجدول Employees باستخدام مفتاح فرعي اسمه EmployeeID مع جعله مفتاحا متفردا Unique:

| ID | USerName | Password | EmployeeID |
|---|---|---|---|
| ١ | ash | .... | ١ |
| ٢ | mgdy | .... | ٢ |
| ٣ | rshdan | .... | ٣ |
| ٤ | ysrhwary | .... | ٤ |

لاحظ أن العمودين ID و EmployeeID في الجدول السابق متماثلان في تركيبهما وبياناتهما، لهذا لا بأس من التخلص من أحدهما (وليكن العمود ID).

## ٤- العلاقة الذاتية Self-Relation:

في بعض الأحيان يمكن إنشاء علاقة بين سجل من سجلات الجدول، وسجل آخر في نفس الجدول.. يسمى هذا بالعلاقة الذاتية Self-Relation، حيث يحتوي الجدول على عمود المفتاح الأساسي وعمود المفتاح الفرعي معا.. هذا مفيد عند تمثيل أي تركيب هرمي كالشجرة Tree.. في هذه الحالة يشير المفتاح الفرعي لكل سجل إلى المفتاح الأساسي للسجل الرئيسي الذي ينتمي إليه.. وعلينا تتبع هذه السلسلة إلى أن نصل إلى سجل رئيسي يعمل كجذر، وفي هذه الحالة يجب أن نضع صفرا أو -١ في خانة المفتاح الفرعي، للإشارة إلى عدم وجود سجل رئيسي لهذا السجل.. افترض مثلا أن عندنا شجرة العائلة التالية:

- كامل
  - شريف
    - أحمد
    - نهى
  - هاني
    - رشيد
    - فتحي
  - عمر
    - نجوى
    - تقوى
    - معاذ

| ParentID | Name | ID |
|---|---|---|
| ٠ | كامل | ١ |
| ١ | شريف | ٢ |
| ٢ | أحمد | ٣ |
| ٢ | نهى | ٤ |
| ١ | هاني | ٥ |
| ٥ | رشيد | ٦ |
| ٥ | فتحي | ٧ |
| ١ | عمر | ٨ |
| ٨ | نجوى | ٩ |
| ٨ | تقوى | ١٠ |
| ٨ | معاذ | ١١ |

يمكن تمثيل هذه الشجرة في جدول واحد كما هو موضح.

في هذا الجدول، يعمل الحقل ID كمفتاح أساسي، بينما يعمل الحقل ParentID كمفتاح فرعي.. وبهذا يكون من السهل استخراج كل التفاصيل المطلوبة لو بدأنا من أي سجل.. مثلا:

- لمعرفة السجلات الفرعية للسجل الحالي، عليك بالبحث في الحقل ParentID عن الخانات التي تحتوي على القيمة الموجود في الحقل ID للسجل الحالي.. مثلا: فتحي ليس له أبناء، لأن معرفه ID = ٧،

غير موجود في الحقل ParentID في أي سجل في الجدول.. بينما عمر له ثلاثة أبناء، لأن معرفه ID = ٨، موجود في الحقل ParentID في السجل رقم ٩ والسجل رقم ١٠ والسجل رقم ١١.

- ولمعرفة السجل الرئيسي للسجل الحالي، يكفي أن نقرأ المعرف ID المحفوظ في الحقل ParentID للسجل الحالي.. مثلا: عمر هو ابن الرجل الذي سجله يحمل المعرف ١، (وهو كامل).

إن هذه أفضل طريقة لتمثيل العلاقات الشجرية، فلو حاولت استخدام أكثر من جدول لتمثيلها، فسيكون عليك إنشاء جدول لكل مستوى من مستويات الشجرة، وإنشاء علاقات بينها، وهو أمر غير عملي ويضيع منك وقتا أطول في تصمم الجداول، خاصة لو ازداد عدد المستويات الفرعية للشجرة، كما أن كتابة الكود الذي يتعامل مع مثل هذه الجداول سيكون مزعجا ومعقدا!.. انظر كيف يمكن تمثيل الشجرة السابقة بثلاثة جداول:

| جدول الجدود Grandparents | |
|---|---|
| **ID** | **Name** |
| ١ | كامل |

| جدول الآباء Parents | | |
|---|---|---|
| **ID** | **Name** | **GrandParentID** |
| ١ | شريف | ١ |
| ٢ | هاني | ١ |
| ٣ | عمر | ١ |

| جدول الأبناء Sons | | |
|---|---|---|
| **ID** | **Name** | **ParentID** |
| ١ | أحمد | ١ |
| ٢ | نهى | ١ |
| ٣ | رشيد | ٢ |
| ٤ | فتحي | ٢ |
| ٥ | نجوى | ٣ |
| ٦ | تقوى | ٣ |
| ٧ | معاذ | ٣ |

طبعا سيكون الأمر كارثيا لو أردت تمثيل عدة أجيال أخرى، كالأحفاد وأبناء الأحفاد وهلم جرا!

## القيود Constrains:

ماذا سيحدث لو تمّ حذف سجل توفيق الحكيم من جدول المؤلفين؟
في هذه الحالة ستظل هناك روايات في جدول الكتب تشير إلى سجل توفيق الحكيم، بينما هو محذوف.. هذا بالتأكيد وضع مثاليّ لحدوث أخطاءٍ مزعجة في برنامجك، لهذا يجبُ أن تراعي منطقيّة وصحّة العلاقات المرجعيّة، وهو ما يعرف باسم التكامل المرجعي بين الجداول **Referential Integrity**، فلو قمت بحذف سجل توفيق الحكيم من جدول المؤلفين، فيجب عليك أيضا أن تحذف كل الروايات والكتب التي ألفها من جدول الكتب.

وللتأكد من فرض هذا التكامل، يمكننا تعريف ما يسمى بالقيود Constrains..
مثلا: لو أضفت قيد المفتاح الأجنبي Foreign-Key Constrain، فإن قاعدة البيانات سترفض حذف أي مؤلف بينما ما تزال هناك كتب تابعة له.. هذا يجبرك على حذف كتب هذا المؤلف أولا قبل أن تقوم بحذفه.

ويوجد قيد آخر نوهنا إليه سابقا، وهو قيد التفرد Constrain Unique، وهو يفرض على المفتاح الأساسي للجدول لضمان تفرد معرف كل سجل.. ويفرض قيد التفرد أيضا على المفتاح الفرعي في حالة واحدة فقط، وهي عند التعامل مع علاقة واحد بواحد One-to-one.

## العروض Views:

العرض هو طريقة لعرض بعض البيانات من جدول أو أكثر.. ويمكن أن تحفظ طريقة العرض هذه في قاعدة البيانات باسم خاص، إذا كنت تحتاج إلى الرجوع إليها أكثر من مرة.
مثلا: يمكنك أن تنشئ في قاعدة بيانات الكتب عرضا اسمه TopSales، يعرض الكتب الأكثر مبيعا.
لاحظ أن العرض لا يستهلك مساحة في قاعدة البيانات، لأن ما يحدث فعليا هو حفظ الاستعلام Query الذي يجمع بيانات هذا العرض، وعند فتح العرض، يتم تنفيذ هذا الاستعلام مرة أخرى لجمع البيانات من الجداول.. هذا يضمن أنك ترى دائما أحدث البيانات الموجودة في الجداول.

وستتعرّف على كل هذه المفاهيم عمليّا، أثناء إنشاء قواعد البيانات، وأثناء استخدامها من خلال برامجنا.

## الفهارس Indices:

افترض أنك تريد أن تبحث عن قيمة معيّنة في أحد الأعمدة، كالبحث مثلا عن (أحمد خالد توفيق) في حقل المؤلفين.. أوّل ما سيجول بذهنك هو أنّ الحاسب سيفحص أسماء المؤلفين واحدا وراء الآخر من البداية إلى النهاية.. ربّما تبدو هذه الطريقة سهلة، ولكنّها أسوأ طريقة يمكن أن يتمّ البحث بها، لأنها قد تتسبّب في بطءٍ شنيع في أداء البرنامج، خاصّة وأنّ الجدول قد يحتوي على عدد كبير من الصفوف، قد يصل عددها إلى عشرات أو مئات الآلاف، وكل خانة من خاناته قد تحتوي على نصوص طويلة، وأنت تعرف أن مقارنة عددين تتم بمقارنة الوحدات الثنائية Bytes في كل منهما (لهذا يحتاج العدد الصحيح int مثلا إلى أربع عمليات مقارنة)، بينما تحتاج مقارنة نصين إلى مقارنة حروف كل منهما، مما يزيد كثيرا من عمليات المقارنة، فقد تحتاج قاعدة البيانات إلى إجراء ٥٠ عملية لمقارنة نصين اثنين طول أقصرهما ٥٠ حرفا بافتراض أسوأ حالة ممكنة، وهي تشابه أول ٤٩ حرفا في النصين!.. هذا مع العلم بأن حقول النصوص قد تحتوي على نصوص أطول من هذا بكثير!

لمثل هذه التحديات، توجد العديد من الطرق لتسريع عملية البحث، تبدأ بترتيب العمود أبجديّا، وذلك لأنّ البحث في عمود مرتب أسرع بكثير من البحث في عمود عشوائيّ، خاصّة مع استخدام طريقة بحث جيّدة، مثل البحث الثنائي Binary Search أو غيره من الطرق.

ولكن، رغم سرعة عملية البحث بعد الترتيب، إلا إنّ الترتيب نفسه قد يكون مأساة!.. هل تتخيّل كم من الوقت يمكن أن يضيّعه برنامجك لترتيب خمسين ألف نصّ مثلا، يتكون كل منها في المتوسط من ٣٠ حرفا؟.. فما بالك إذن حينما يقوم البرنامج بتكرار البحث في نفس العمود أكثر من مرّة؟!

إذن فما الحل؟

الحل هو أن نرتب بيانات العمود في قاعدة البيانات منذ البداية، وبهذا نضمن سرعة البحث، دون الحاجة إلى تكرار عملية الترتيب في كل مرة.. لكن هذا يقودنا إلى سؤال آخر: وماذا لو كان الجدول يحتوي على أكثر من عمود تحتاج للبحث فيها، فتبعا لأيّ منها سوف نرتب الجدول يا ترى؟!!

هنا يبرز حل رائع اسمه الفهارس Indexes:

تخيّل أننا طلبنا من قاعدة البيانات إنشاء فهرس لعمود المؤلفين في جدول الكتب.. هنا سيتم ترتيب هذا العمود، مع إنشاء فهرس يحتفظ ببيانات توضح ترتيب كل مؤلف.. فمثلا، إذا كان توفيق الحكيم هو المؤلف رقم ١ في الجدول، ونبيل فاروق هو المؤلف رقم ٧، وأحمد خالد توفيق هو المؤلف رقم ١٠، فإن الفهرس سيحتفظ بمعلومات تفيد بأن أول مؤلف في الترتيب الأبجدي يوجد في الصف رقم

١٠، يليه في الترتيب المؤلف الذي يقع في الصف رقم ١، يليه المؤلف الذي يقع في الصف رقم ٧.. هذا يحقق لنا الفوائد التالية:

- يمكن استخدام هذا الفهرس لتطبيق خوارزميات البحث على عمود المؤلفين وكأنه عمود مرتب، وبالتالي ستكون عمليّة البحث أسرع.
- لن نحتاج إلى إجراء أي تغيير في طريقة عرض البيانات في قاعدة البيانات، حيث سيظل ترتيب جدول المؤلفين في الجدول كما هو، وبالتالي سنترك للمستخدم حريّة ترتيب البيانات كما يحلو له.
- يمكننا إنشاء أكثر من فهرس لأكثر من عمودٍ في نفس الجدول، ممّا يعني قدرتنا على ترتيب كل هذه الأعمدة والبحث فيها.
- سنوفر وقت ترتيب العمود أثناء البرمجة، وإن كان هذا الوقت سيستهلك عند إنشاء الفهرس لأول مرّة في وقت التصميم.. لاحظ إن إنشاء الفهرس والجدول فارغ أو به القليل من الصفوف، يجعل عملية إنشاء الفهرس أسرع، وكلما أضفت سجلا جديدا إلى الجدول، يتم تحديث الفهرس لوضع هذه السجل في ترتيبه في الفهرس.

لاحظ أنك تستطيع إنشاء فهرس لأكثر من عمود معا، حيث سيتم الترتيب تبعا للعمود الأول، فإن تماثلت قيم بعض الصفوف (مؤلفان يشتركان في الاسم الثلاثي مثلا)، يتم ترتيب هذه الصفوف على أساس العمود الثاني (اسم الدولة مثلا).

| **ملحوظة:** |
| --- |
| بمجرد جعل الحقل مفتاحا أساسيا للجدول، يتم إنشاء فهرس له، نظرا لأن المفتاح الرئيسي يستخدم بغزارة في الاستعلامات التي تبحث عن سجلات معينة الجدول، خاصة مع استخدامها في علاقات مع جداول أخرى. |

ولكن.. للأسف لا توجد طريقة مثاليّة، فهناك عيبان أساسيّان للفهارس:

١- الفهارس المنشأة يتمّ حفظها في قاعدة البيانات، ممّا يعمل على زيادة حجمها.. لهذا إذا كان حجم قاعدة بياناتك كبيرا، فلا بدّ أن توازن بين حاجتك إلى توفير الوقت وحاجتك إلى توفير الحجم.. لكني لا أظن هذه مشكلة كبيرة اليوم، مع السعات الهائلة لوسائط التخزين.

٢- عند تعديل قيمة أيّ خانة في العمود المفهرس، أو حذف أو إضافة أيّ صفّ، لا بدّ من إعادة تحديث الفهارس لمراعاة الاختلاف الذي أحدثه هذا التغيير.. مثل هذا الأمر يسبّب عيبا خطيرا، هو بطء عمليّات التعديل والحذف والإضافة بصورة ملحوظة، تزداد سوءا مع ازدياد طول الجدول، ومع تكرار هذه العمليات في برنامجك بشكل كبير.. هنا يجب أن توازن في برنامجك، بين حاجتك إلى سرعة البحث وحاجتك إلى سرعة التحديث والحذف والإضافة.. لديك هذه الحالات:

أ- إذا كان برنامجك يقوم بمئات من عمليّات التحديث والإضافة والحذف بطريقة متتالية ومن عدد كبير من المستخدمين، بينما لا توجد الكثير من عمليات البحث، فلا داعي لإنشاء الفهرس!

ب- إذا كنت تستخدم قاعدة بيانات يتعامل معها المستخدمون في معظم الوقت لقراءة البيانات وإجراء الاستعلامات وعمليات البحث، فإن الفهرس سيجعل هذه العمليات أسرع بكثير، ومع قلة عمليات التحديث والإضافة والحذف (المتاحة فقط لمدير النظام)، لن يكون عبء الفهارس ملحوظا.

ج- إذا كنت تحتاج إلى كلا النوعين من العمليات في نفس الوقت، فيمكنك تقليل العبء الناتج من الفهارس في برنامجك بحيلة صغيرة: فعندما تكتب في الكود حلقات التكرار Loops أو الاستعلامات التي تقوم بتعديل أو إضافة أو حذف مئات أو آلاف السجلات على التوالي، فاحذف الفهرس من الجدول قبل بدأ تنفيذ هذا الكود، ثمّ قم بإعادة إنشائه مرّة أخرى بعد انتهاء هذه العمليات، وبهذا تقلل من عبئه وتستفيد منه في تسريع عمليات البحث في نفس الوقت!

لكن عليك ملاحظة أن هذه الحيلة ستكون خطيرة إذا كان هناك عدد كبير من المستخدمين يتعاملون مع قاعدة البيانات في نفس اللحظة، وبعضهم يبحث وبعضهم يحذف السجلات!!.. لهذا فأنت الوحيد الذي يعرف طبيعة عمل البرنامج وأنسب اختيار لجعله أكفأ ما يمكن.

**وكقاعدة:** أنشئ فهارس للحقول النصية التي تتوقع أن يحتاج مستخدمو برنامجك للبحث فيها بغزارة.

-٢-

# قواعد بيانات Access

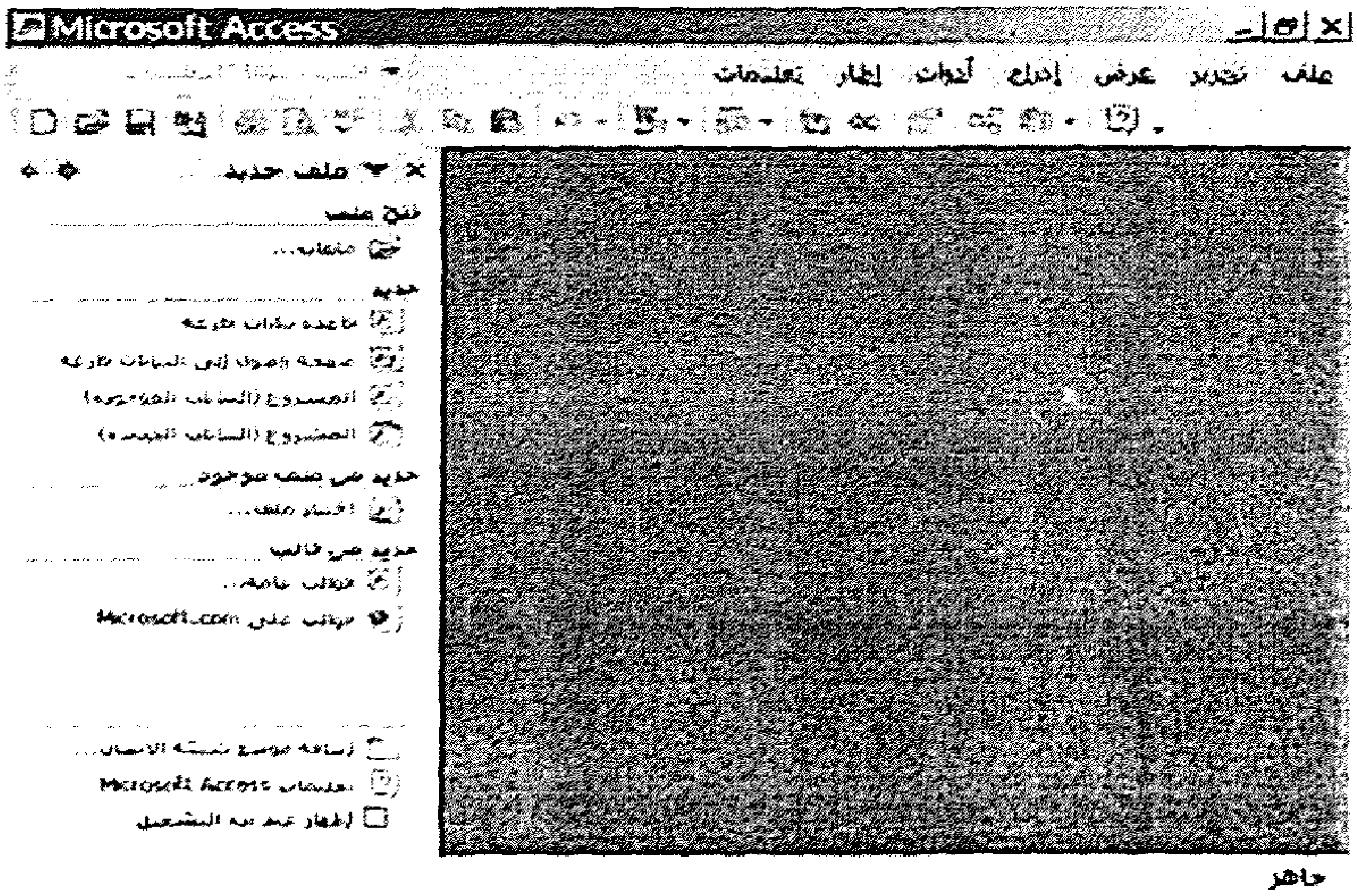
لن تستطيع التعامل مع قواعد البيانات، ما لم تكن لديك المقدرة على إنشاء قواعد البيانات وما تحتويه من جداول.. تعالَ نأخذ فكرةً سريعةً عن هذا الأمر.

## إنشاء الجداول بتطبيق Access:

Access هو أحد تطبيقات Microsoft Office.. لن تجد فارقا يذكر بين إصدارات Access 2000 و Access XP و Access 2003.. لكنك ستجد اختلافا كبيرا في واجهة الاستخدام بدءا من Access 2007.. دعنا هنا نتعامل مع Access XP وعليك التعامل مع الفوارق البسيطة بين الإصدارات المختلفة بنفسك.

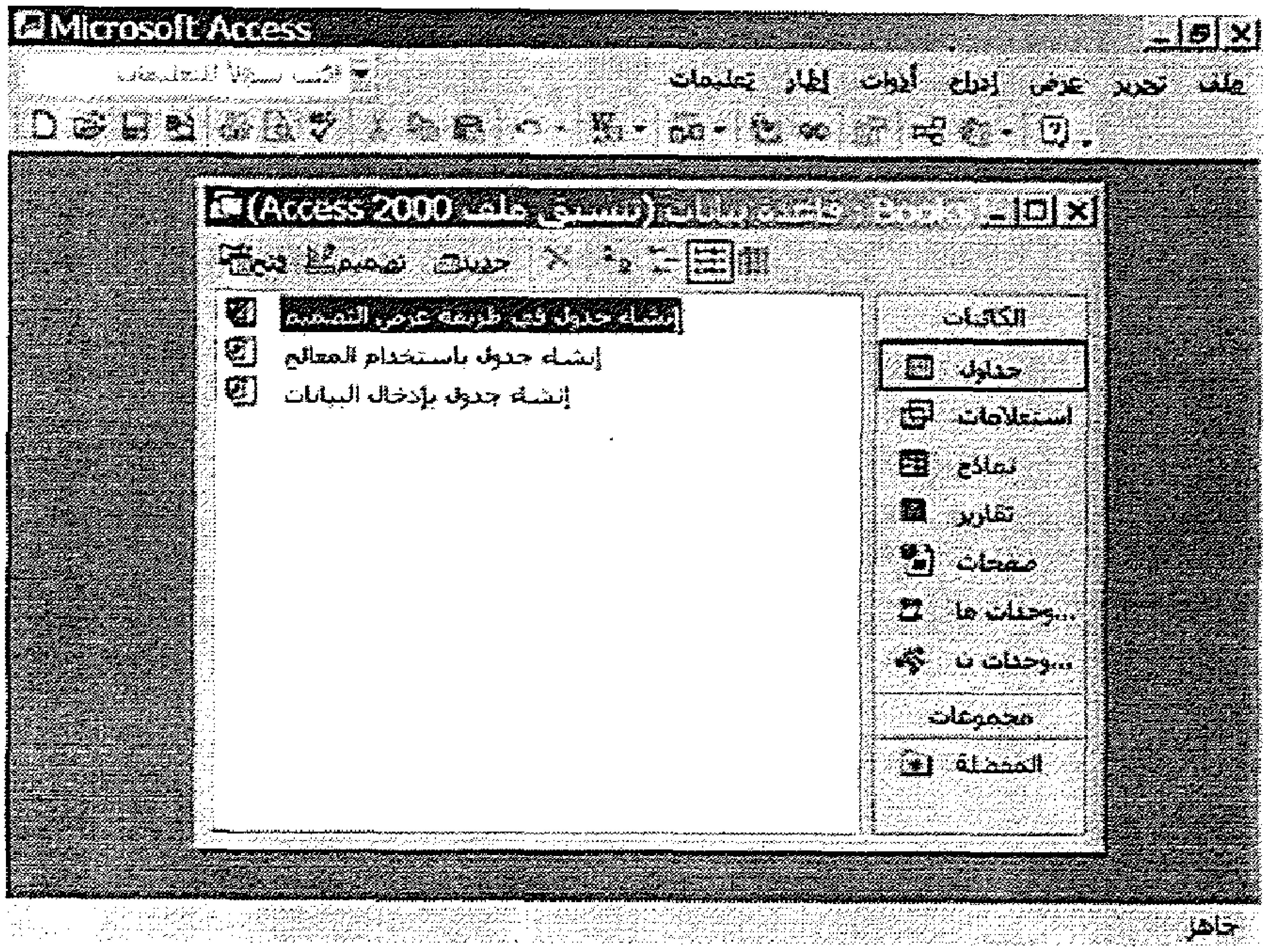
افتح تطبيق Access ومن القائمة "ملف" File اختر "جديد" New.. أنا أكتب أسماء القوائم بالعربية والإنجليزية لأن البعض لديه واجهة استخدام عربية، والبعض لديه واجهة إنجليزية، على حسب النسخة التي أعدها على جهازه.

ستظهر لك نافذة "ملف جديد" New File في جانب واجهة التطبيق كما في الصورة التالية:

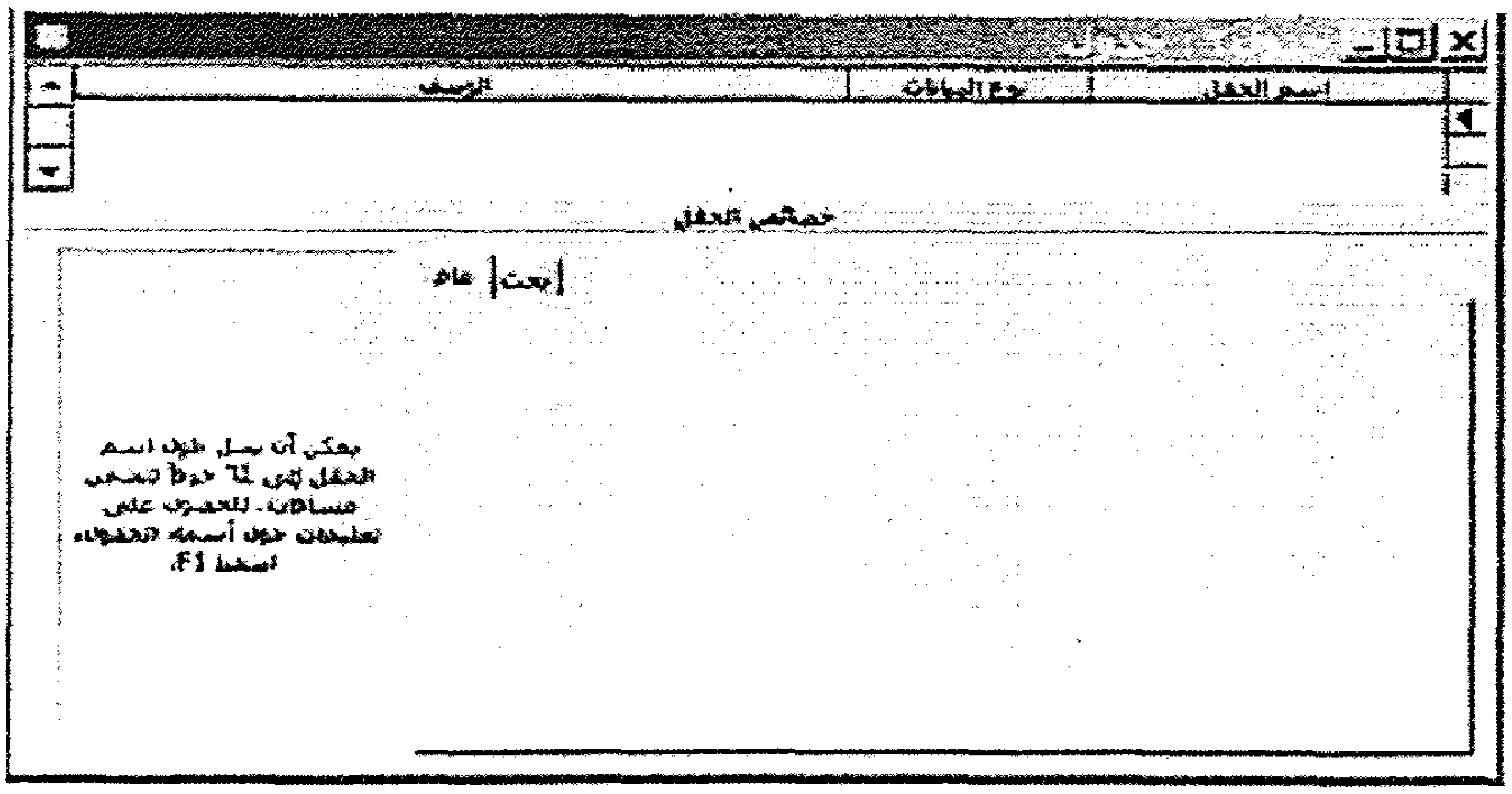


تحت الشريط "جديد" New اضغط الأمر "قاعدة بيانات فارغة" Blank Database.. سيظهر لك مربّع حوار "حفظ ملف" Save-File Dialog، وهو يطلب منك تحديد اسم قاعدة البيانات وموقع حفظها على الجهاز.. غيّر اسم قاعدة البيانات من الاسم الافتراضي إلى "Books.mdb"، وقم بحفظها في الموضع الذي تريده.

بعد إغلاق مربّع الحوار ستظهر لك نافذة قاعدة البيانات كما في الصورة التالية:




انقر الأمر "إنشاء جدول في طريقة عرض التصميم" Create Table in Design View مرّتين بالفأرة.. ستظهر لك نافذة تصميم الجدول كما في الصورة التالية:



كل ما عليك الآن، هو كتابة أسماء الحقول وتحديد نوع البيانات التي ستضعها في كل منها.. تعالَ نبدأ بتصميم جدول أسماء المؤلفين.. هذا الجدول سيتكوّن من الحقول التالية:

| | اسم الحقل Field Name | نوع البيانات Data Type | الوصف Description |
|---|---|---|---|
| | ID | (ترقيم تلقائي) AutoNumber | المفتاح الرئيسيّ لهذا الجدول |
| | Author | Text (نص) | اسم المؤلف |
| | CountryID | (رقم) Number | رقم الدولة التي ينتمي إليها المؤلف. |
| | Phone | Text (نص) | رقم هاتف المؤلف. |
| | About | (مذكرة) Memo | نبذة عن المؤلف |

| نصيحة: |
|---|
| سمّ الجداول والأعمدة بأسماء أجنبيّة، لأنّ هذه الأسماء ستدخل في كتابة الكود في فيجيوال بيزيك، وسيكون من المرهق الانتقال من الإنجليزية إلى العربيّة والعكس أكثر من مرّة أثناء كتابة الكود. |

بعد أن تنشئ هذه الحقول، قم بما يلي:

- اضغط بزر الفأرة الأيمن على المربّع الرماديّ المجاور للحقل ID، ومن القائمة الموضعيّة Context Menu اضغط الأمر "مفتاح أساسي" Primary Key لجعل هذا الحقل مفتاحا أساسيًا للجدول.. ستلاحظ ظهور رمز المفتاح أمام هذا الحقل.. ستلاحظ كذلك أنّ الخاصيّة "مفهرس" Indexed في المنطقة السفلى من النافذة ستتحوّل إلى "نعم بدون تكرار" Yes (No Duplicates)، وهو شيء متوقع، فلقد اتفقنا أنّ قيم المفتاح الأساسيّ غير قابلة للتكرار.
- اضغط بالفأرة على حقل اسم المؤلف Author، وفي المنطقة السفلى من النافذة غيّر قيمة الخاصيّة "حجم الحقل" Field Size إلى ٣٠ بدلا من ٥٠، وذلك بافتراض أن أطول اسم مؤلف لن يزيد عن ٣٠ حرفا.
- اضغط بالفأرة على الحقل CountryID، وفي المنطقة السفلى من النافذة غيّر قيمة الخاصيّة "حجم الحقل" Field Size إلى Byte.. السبب في هذا أننا لن نتعامل إلا مع ٢٢ دولة عربية فقط.. وحتى لو أضفت بعض البلاد الأجنبية، فإن الوحدة الثنائية Byte تقبل أرقاما من ٠ إلى ٢٥٥، وهو ما

يستوعب أهم أسماء دول العالم.. لاحظ أن اختيار حجم أقل للحقل يوفر مساحة قاعدة البيانات، ويجعل التعامل معها أكفأ.. لكن عليك أن تراعي أي توقعات مستقبلية للتوسع، فمحاولة تكبير حجم الحقل بعد ذلك تستغرق وقتا ملموسا في قواعد البيانات الضخمة، وهو ما قد يؤدي إلى تعطيل الموظفين الذين يتعاملون مع قاعدة البيانات في ذلك الوقت.. لهذا ينصح بإجراء التعديلات على تركيب قاعدة البيانات في الأوقات التي يقل فيها تعامل الموظفين والعملاء معها.

- اضغط بالفأرة على الحقل Phone، وفي المنطقة السفلى من النافذة غيّر قيمة الخاصيّة "حجم الحقل" Field Size إلى ٢٠.. أعرف أنك تتساءل لماذا لم نجعل رقم الهاتف رقما.. في الحقيقة هناك عدة أسباب:

١- حفظ رقم الهاتف كنص يتيح لك وضع بعض المسافات وعلامات الترقيم المميزة فيه، مثل:

+2 - 0102020375
+2 – 050 – 17354-036

هذا يجعله أكثر وضوحا عند قراءته.. لكن البعض يرد على هذه النقطة قائلا إن من الممكن حفظ رقم الهاتف في قاعدة البيانات كرقم، وعند عرضه في نموذج فيجيوال بيزيك يمكن تنسيقه بأي شكل مطلوب.. من وجهة نظري أن هذا قد يسبب بعض المشاكل، بسبب اختلاف تركيب رقم الهاتف من دولة إلى أخرى.

٢- نظرا لوجود مفتاح لكل دولة، ومفتاح لكل منطقة، فإن رقم الهاتف سيكون عددا كبيرا جدا، ولن يكفيه العدد الصحيح الطويل Long Integer (وهو يستخدم 4 Bytes)، وهو ما يتطلب استخدام نوع بيانات كبير، وفي هذه الحالة لن يكون هناك توفير كبير عن استخدام النص!

٣- أهم نقطة في الأمر، هو أن تطبيق آكسيس يحذف الأصفار المكتوبة على يسار أي رقم، لهذا لو كتبت 002010202375 فسيحولها إلى 2010202375، وهو ما سيجعل من الصعب عليه معرفة مفتاح الدولة!

٤- إذا أردنا حل مشاكل استخدام الأرقام، فلا بد من حفظ مفتاح كل دولة بمفرده في عمود خاص به.. في هذه الحالة يجب وضعه في جدول أسماء الدول وربطه بجدول المؤلفين.. وربما نحتاج إلى فعل نفس الشيء مع مفتاح كل منطقة (مفتاح المحافظة).. فعل هذا قد يجعل الأمور أعقد برمجيا ويضيع وقتا دون أن يوفر مساحة تذكر.

الآن اضغط Ctrl+S لحفظ الجدول.. سيظهر لك مربّع إدخال يطالبك بإدخال اسم الجدول.. سمّه Authors واضغط زر OK.

الآن لو أغلقت نافذة التصميم، فستجد الجدول Authors قد أضيف إلى نافذة قاعدة البيانات.. انقره مرّتين بالفأرة.. سيؤدي هذا إلى فتح نافذة جديدة تعرض لك جدول المؤلفين في وضع التحرير، حيث يمكنك إدخال بيانات كل مؤلف، مع ملاحظة أن حقل الترقيم التلقائي ID لا يسمح لك بالكتابة، فقيمته تزيد تلقائيا كلما أدخلت سجلا جديدا.

| ملاحظات حول تحرير الجداول: |
| --- |
| - ضغط رأس العمود بالفأرة، يحدد العمود كله.<br>- ضغط رأس الصفّ بالفأرة، يحدد الصفّ كله.<br>- يمكن الانتقال من خانة إلى أخرى بضغط الزرّ Tap.<br>- لإدخال سجل جديد اكتب في السجل الفارغ في نهاية الجدول.<br>- عند تغيير قيمة أيّ خانة، لا يتمّ حفظ التغييرات إلا عند الخروج من الصف الذي توجد فيه إلى صف آخر، أو عند ضغط زر الحفظ.. فإذا وجد آكسيس أن إحدى القيم غير مقبولة، عاد إلى نفس الصف وأعلن رفضه للتغييرات التي قمت بها، ولا يسمح لك بمغادرة الصف قبل تصحيح القيمة الخاطئة.<br>- لحذف أيّ صفّ، حدّده واضغط الزر Delete من لوحة المفاتيح، أو اضغط أمر Delete من القائمة الموضعية Context Menu. |

وبنفس الطريقة يمكنك تصمّم باقي الجداول.. ستجد هذه الجداول في قاعدة البيانات Books.mdb المرفقة بأمثلة هذا الكتاب، وستجد فيها الجداول التالية:

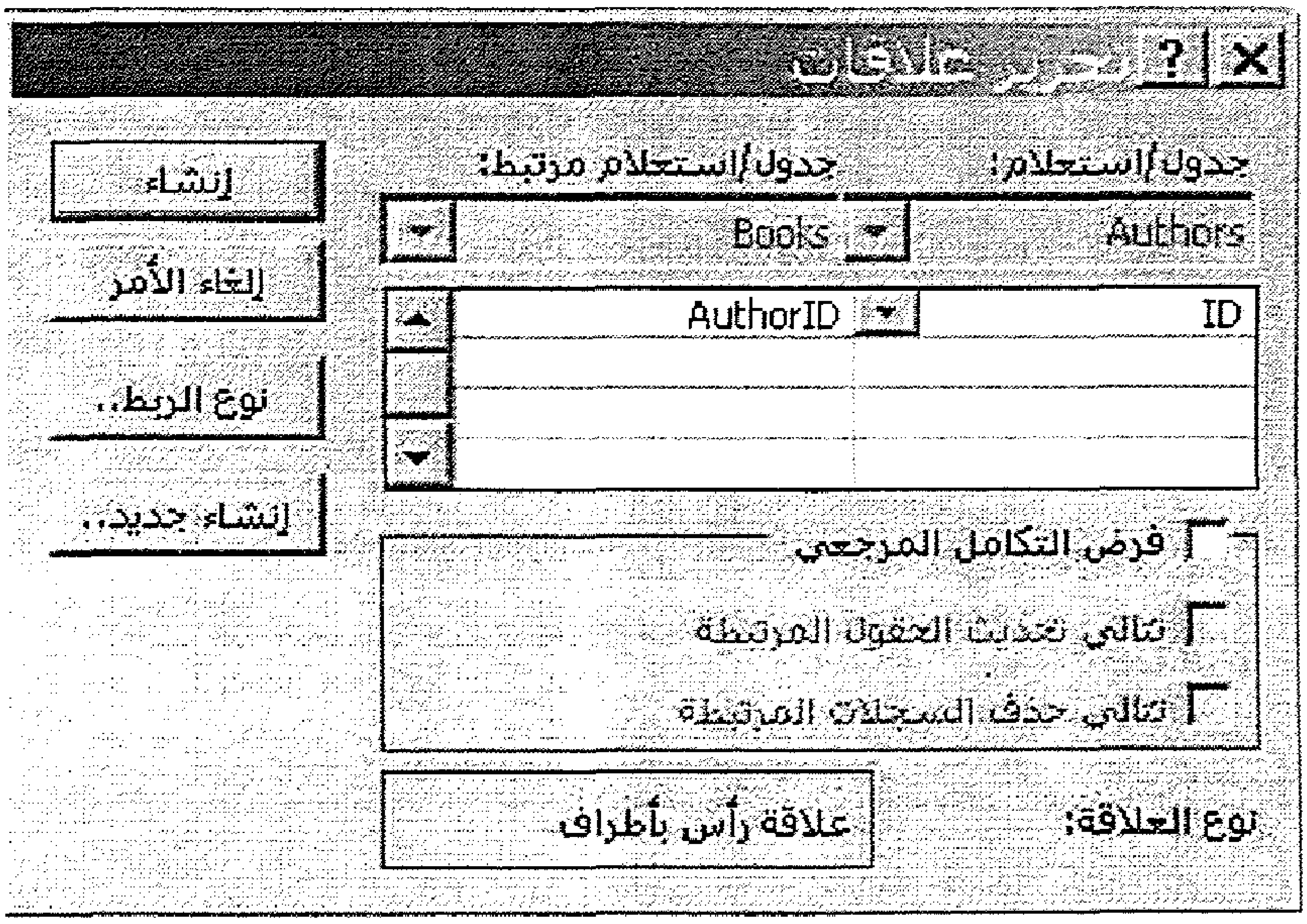
| | |
| --- | --- |
| Authors | جدول المؤلفين. |
| Books | جدول الكتب.. لاحظ أننا سنحفظ في هذا الجدول رقم الناشر ورقم المؤلف وعدد النسخ الكلية المتاحة من كل كتاب، وسعر النسخة، وتاريخ طباعة الكتاب، ورقم الطبعة. |
| Classes | جدول أصناف الكتب، وهو مرتبط بجدول الكتب. |
| Countries | أسماء الدول، وهو مستخدم مع جدولي المؤلفين والناشرين. |
| Publishers | جدول الناشرين. |
| Sales | جدول مبيعات الكتب، وهو يحتوي على فواتير بيع الكتب، حيث سنسجل عدد النسخ المبيعة من كل كتاب، والخصم الممنوح للمشتري، واسم المشتري، وتاريخ البيع. |

## إنشاء علاقات بين الجداول:

دعنا الآن نربط حقلي رقم المؤلف بين جدولي المؤلفين والكتب، لتحقيق التكامل المرجعيّ Referential Integration بين الجدولين:

من قائمة الأدوات Tools اضغط الأمر "علاقات" Relationships.. ستظهر لك نافذة "إظهار جدول" وفيها أسماء كل جداول قاعدة البيانات.. اختر الجدول الذي تريد استخدامه في العلاقة واضغط زرّ الإضافة Add، وكرر هذا مع كل الجداول المطلوبة.. بعد أن تنتهي أغلق هذه النافذة، لتظهر لك نافذة العلاقات، وفيها ستجد كل جدول في صورة مستطيل عليه عنوان الجدول، وبه أسماء أعمدته.

ولإنشاء علاقة بين الجدولين، اسحب الحقل ID بالفأرة من جدول المؤلفين، وتحرّك بالفأرة إلى الحقل AuthorID في جدول الكتب.. ستلاحظ تغيّر شكل مؤشّر الفأرة.. اترك زر الفأرة الأيسر فوق هذا الحقل.. هنا ستظهر لك نافذة إنشاء العلاقة بين الحقلين:




اضغط الاختيار "فرض التكامل المرجعي" Enforce Referential Integrity، لتقوم قاعدة البيانات آليّا بالتحقق من صحّة البيانات بين الجدولين.. وفي هذا الصدد لديك اختياران:

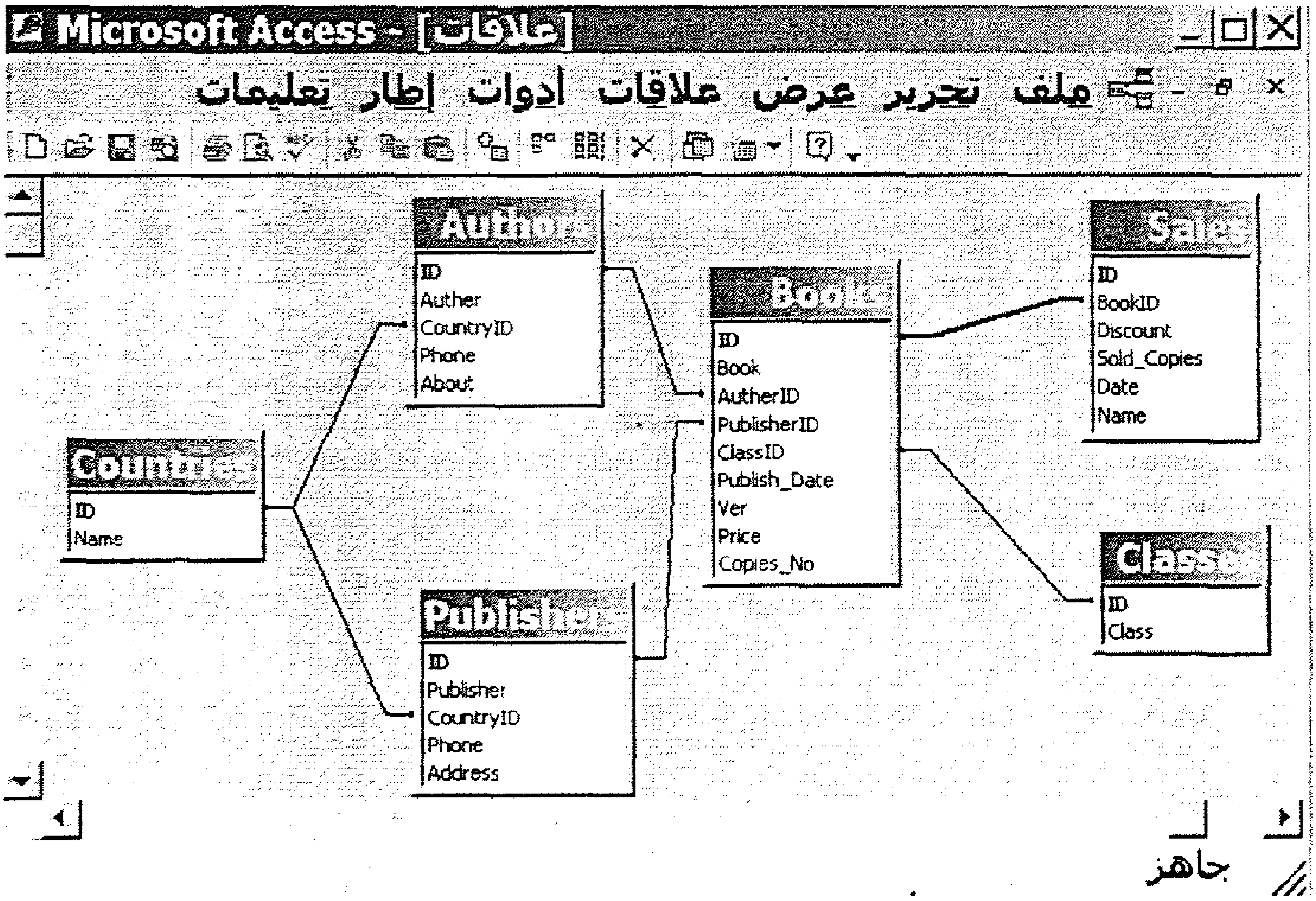
- **تتالي تحديث الحقول المرتبطة Cascade Update Related Fields:**

فمثلا، لو غيّرت رقم المؤلف (توفيق الحكيم) من ١ إلى ١٠، فسيتمّ تغيير رقم المؤلف AuthorID تلقائيّا ليصير ١٠، في سجلات كل الكتب التي ألفها.

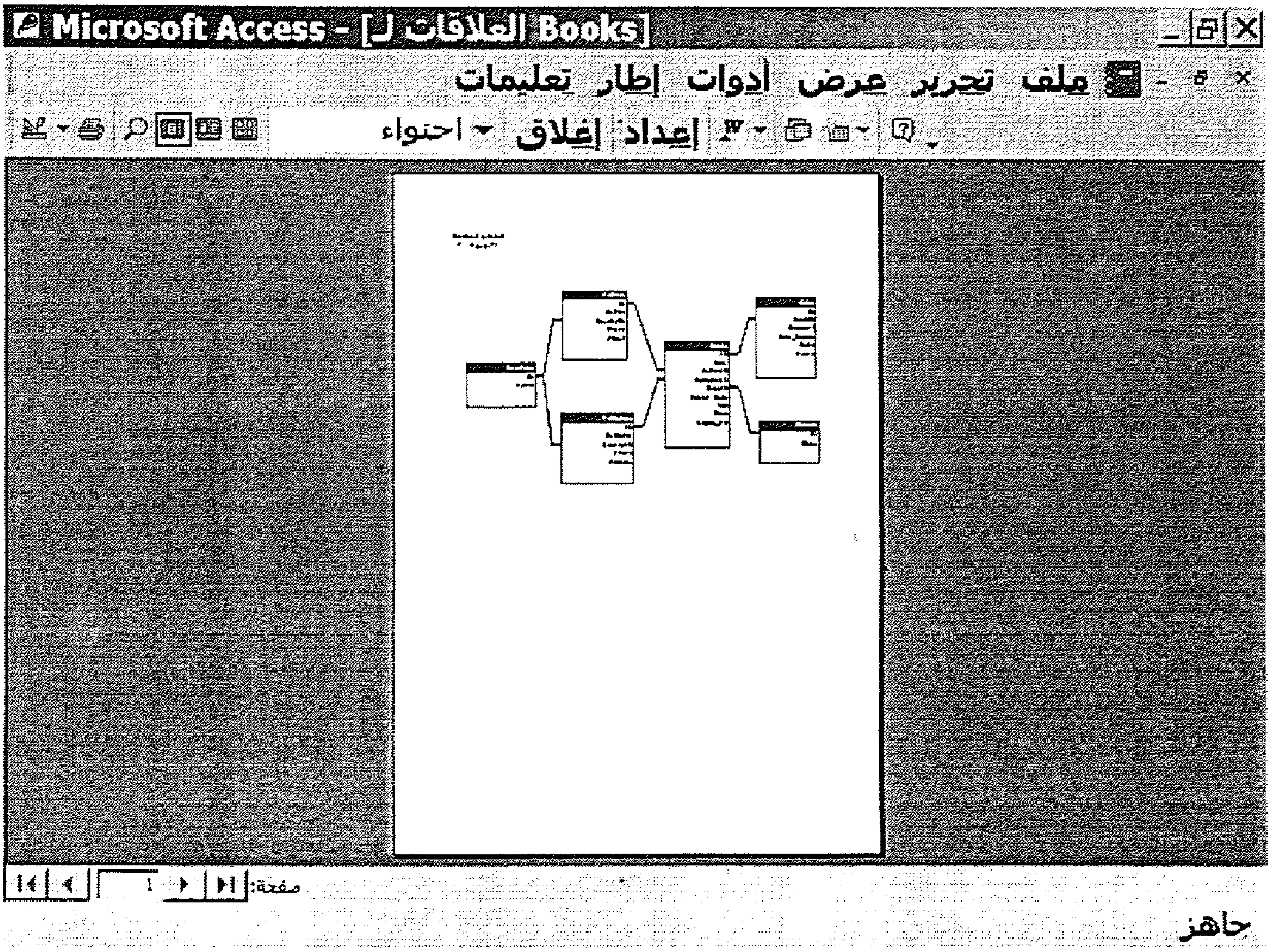
| ملحوظة: |
| --- |
| لن تستطيع تغيير قيمة رقم المؤلف ID في جدول المؤلفين، لأنّه ترقيم تلقائيّ.. ولو أردت تغييره، فعليك بتغيير نوع الحقل من ترقيم تلقائي إلى رقم، على أن يكون هذا قبل إنشاء العلاقة، فمن غير المسموح تغيير نوع بيانات حقل داخل في علاقة.. فإذا كنت قد أنشأت العلاقة فعلا، فعليك حذفها أولا ثم تغيير نوع الحقل، ثم إعادة إنشاء العلاقة مجددا! |

- **تتالي حذف الحقول المرتبطة Cascade Delete Related Fields:**
فمثلا، لو حذفت سجل المؤلف (نبيل فاروق) من جدول المؤلفين، فسيتمّ حذف كل كتب (نبيل فاروق) تلقائيّا من جدول الكتب.

اضغط الزرّ "إنشاء" Create لإغلاق هذه النافذة.. الآن سيظهر خطّ يربط الجدولين.. ولو أردت حذف أي علاقة، فاضغط الخط الخاص بها بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اختر "حذف" Delete.
بمثل هذه الطريقة يمكنك إنشاء العلاقات بين جداول قاعدة بيانات الكتب، كما هو موضح في الصورة:

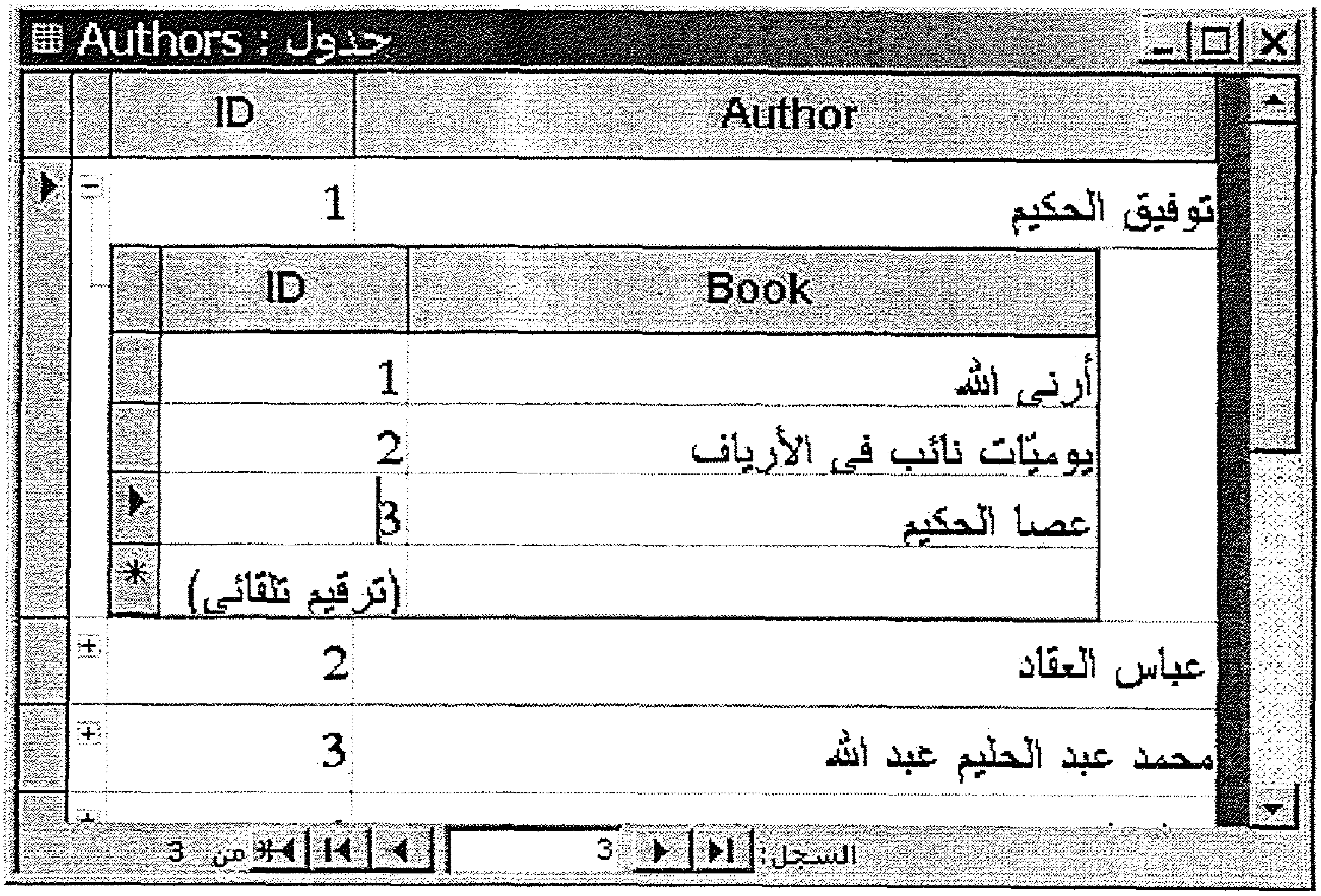


وكما ترى، تقدم لك نافذة العلاقات مخططا بيانيا يلخص كل تفاصيل قاعدة البيانات والعلاقات التي تربط بين جداولها، مما يسهل عليك مراجعة تصميمها في أي لحظة، كما يمكنك طباعة هذا المخطط باستخدام الأمر "طباعة العلاقات" Print Relations من القائمة File، حيث سيؤدي هذا إلى عرض نافذة جديدة بها مخطط العلاقات في شكل صفحة قابلة للطباعة:




في هذه النافذة يمكنك استخدام الأمر "إعدادات الطباعة" Rrint Setup و "طباعة" Print من القائمة File لطباعة الصفحة المعروضة.

قم بحفظ التغييرات، وأغلق نافذة العلاقات.. افتح الآن جدول المؤلفين.. ستلاحظ ظهور العلامة "+" بجوار كل حقل.. اضغط أيّا منها، ولتكن المجاورة لـ (توفيق الحكيم).. ستجد أنّ العلامة "+" تحولت إلى العلامة "–"، وأنّ جدولا صغيرا يحتوي على الكتب التي ألفها توفيق الحكيم قد ظهر.. ولإخفائه ثانية أعد ضغط العلامة "–".




ويمكنك إدخال المزيد من كتب (توفيق الحكيم) في هذا الجدول الفرعيّ، دون أن تُدخل رقمه في الخانة AuthorID في كل مرّة، إذ ستتمّ كتابة هذا الرقم آليّا، ممّا يوفر عليك الوقت ويعفيك من احتمالات الخطأ.

## إنشاء قيد Constraint:

افترض أنك لسبب ما تريد أن تمنع مُدخل البيانات من كتابة مؤلف اسمه (موهوب) في حقل أسماء المؤلفين.. لفعل هذا افتح تصميم جدول المؤلفين، وحدّد الحقل Authors.. ستجد ضمن الخصائص الموجودة في المنطقة السفليّة من النافذة خاصيّة تسمّى "قاعدة التحقق من الصحّة" Check Constraint.. هذا هو المكان الذي سنظلم فيه المؤلف العبقريّ (موهوب) ونحرمه من حقّه المشروع في الظهور في قائمة المؤلفين!!

اكتب في هذه الخانة ما يلي:

<> "موهوب"

اضغط زرّ الحفظ.. ستظهر لك رسالة تستوثق من رغبتك في أداء هذه القاعدة، التي ربّما تستهلك بعض الوقت نتيجة فحص أسماء المؤلفين أولا للتأكد من أنها تخضع لهذا القيد.. اضغط OK.

افتح تخطيط الجدول، وحاول أن تكتب الاسم (موهوب) في حقل المؤلفين.. تذكر أنّ التغيير لن يحدث قبل أن تغادر الصف الذي تكتب فيه.. وفور أن تفعل هذا،

ستظهر لك رسالة تخبرك أنّ إحدى القيم المدخلة غير مسموح بها، وسيعود المؤشّر إلى تلك الخانة مرة أخرى!.. غيّر هذا الاسم، فمهما حاولت لن تستطيع إقناع قاعدة البيانات بقبوله.
هذه الإمكانيّة مفيدة، وتمكنك من وضع شروط معقدة تدخل فيها بعض دوال فيجيوال بيزيك المسموح باستخدامها في جمل SQL.. ويمكنك أن تضغط الزرّ المجاور لخانة التحقق من الصحّة لتظهر لك نافذة تساعدك في بناء الشرط، حيث تعرض لك كل الدوال والتعبيرات المسموح باستخدامها.

## ضغط قاعدة البيانات Database Compacting:

عندما تحذف أحد السجلات من أحد الجداول، فإنه يختفي من عرض الجدول، ولكنه في الواقع يظل في موضعه في ملفّ قاعدة البيانات.. كل ما حدث هو أنّ قاعدة البيانات قد أشارت إلى حالته باعتباره سجّلا محذوفا، حتى لا يتمّ التعامل معه.. ولو حدث في برنامجك أن أصبح السجل الذي تتعامل معه محذوفا (بسبب قيام مستخدم آخر لقاعدة البيانات بحذفه)، فإنّ محاولة قراءة البيانات أو كتابتها في هذا السجل ستؤدّي إلى حدوث خطإ في برنامجك.
وقد تظن أنّ هدف إخفاء السجل المحذوف هو منحك القدرة على التراجع عن حذفه.. ولكنّ هذا ليس صحيحا على الإطلاق، فالسجل الذي يحذف لا يمكن استعادته، رغم أنه ما زال موجودا في ملفّ قاعدة البيانات موسوما بأنه محذوف!
لماذا إذن تستخدم قاعدة البيانات هذه الطريقة؟
تخيّل أنك تريد حذف سجل موجود في منتصف ملفّ قاعدة البيانات.. كيف في نظرك يحدث هذا؟
لحذف هذا السجل يجب أن ينشئ تطبيق قاعدة البيانات ملفا احتياطيّا، وينسخ إليه البيانات السابقة للسجل المراد حذفه، ثمّ البيانات التالية له، وبهذا يتكوّن ملفّ جديد لا يحتوي على السجل المحذوف.. بعد هذا يجب أن يحذف التطبيق ملفّ قاعدة البيانات القديم، ثمّ يسمّى الملفّ الجديد باسم قاعدة البيانات.. واضح أنّ هذه الطريقة ستؤدي إلى بطء عمليّة الحذف، بسبب الاضطرار إلى نسخ كل محتويات ملفّ قاعدة البيانات لمجرد حذف سجل واحد.. تخيل حجم الكارثة التي ستحدث لو حاولت حذف ألف سجل على التوالي بهذه الطريقة!
لهذا كان الحل هو أن يتمّ ترك السجل المحذوف في موضعه، على أن تتمّ الإشارة إلي حالته باعتباره سجلا محذوفا.
ولكنّ معنى هذا أنّ حجم قاعدة البيانات يمكن أن يتضخّم باستمرار مع استمرار العمل عليها والإضافة إليها، لأنّ كل ما يحذف منها يظل في موضعه.. لدرجة أنك لو حذفت كل السجلات من كل الجداول، فسيظل لقاعدة البيانات نفس حجمها!!

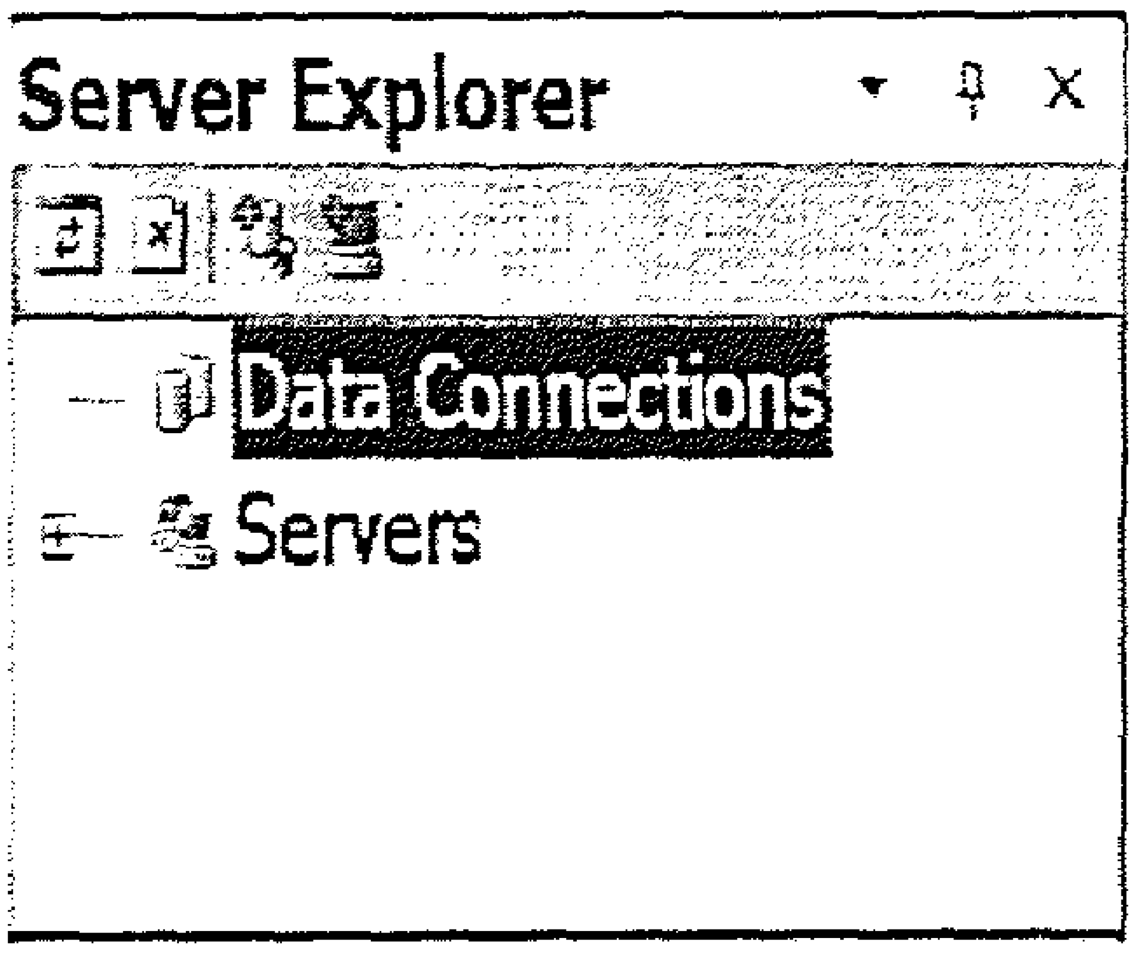
ولحل هذه المشكلة، يجب أن تقوم بين فترة وأخرى بعملية تسمى "ضغط قاعدة البيانات" Database Compacting، وفيها يتمّ إنشاء نسخة جديدة من ملفّ قاعدة البيانات لا تحتوي على السجلات المحذوفة.
وللقيام بهذا في Access، اضغط قائمة الأدوات Tools، ومنها اضغط القائمة الفرعيّة "أدوات مساعدة لقواعد البيانات" ومنها اضغط الأمر "ضغط قاعدة بيانات وإصلاحها" Compact and repair database.. هنا سيقوم Access بإغلاق قاعدة البيانات وضغطها ثمّ إعادة فتحها.. ولو اختبرت حجم ملفّ قاعدة البيانات، فربّما تجده أقل مما كان عليه.

***

بهذه النبذة السريعة، نكون قد أنهينا حديثنا عن Access.. وطبعا ليست هذه هي كل إمكانيّات هذا التطبيق الشهير، فهو قادر في حدّ ذاته على إنشاء النماذج والتقارير، وكتابة كود لغة خاصة من فيجيوال بيزيك تسمّى Visual Basic For Applications أو اختصارا VBA، وهو ما يخرج عن نطاق هذا الكتاب.. إنّ Access له وحده مراجع متخصصة!
والآن، دعنا نتعرف على كيفية الاتصال بقاعدة بيانات Access في دوت نت.

## متصفح الخوادم Server Explorer:

توجد في فيجيوال ستديو نافذة اسمها متصفح الخوادم Server Explorer، وهي تحوي كل الأدوات الأساسيّة اللازمة للاتصال بقاعدة البيانات والتعامل معها.. ويمكنك عرض هذه النافذة بضغط الأمر Server Explorer من القائمة View.. وعندما تظهر لك نافذة متصفح الخادم، ستجد فيها عنصرين:




**روابط البيانات Data Connections:**

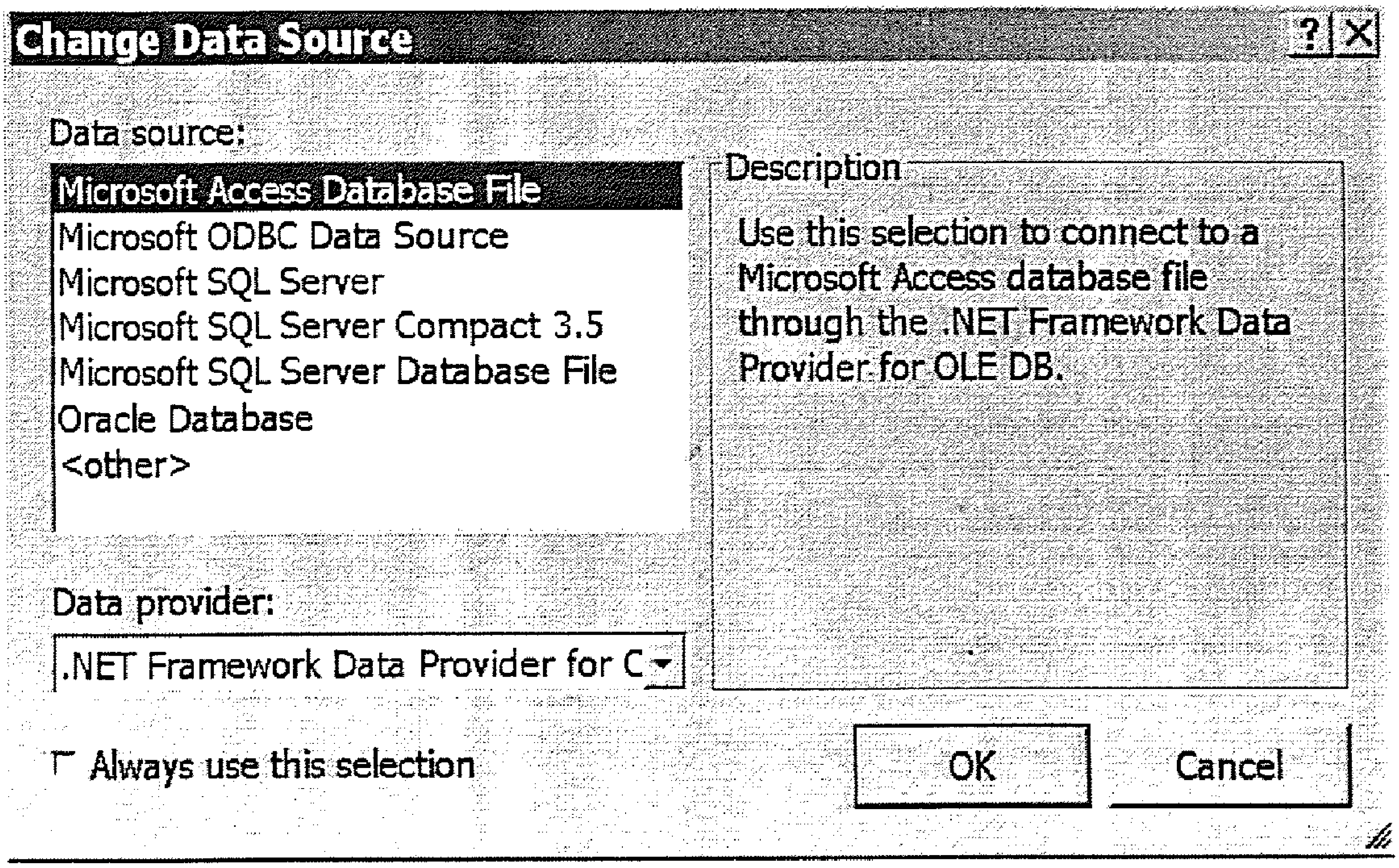
تحت هذا العنصر، تظهر أسماء قواعد البيانات التي تتعامل معها في تطبيقاتك المختلفة.. وسنرى كيف يمكن الاتصال بقواعد البيانات لاحقا.

**الخوادم Servers:**

تحت هذا العنصر، تظهر أسماء خوادم قواعد البيانات Database Servers التي تتعامل معها، وما تمنحه لك من كائنات.. وستجد مبدئيا اسم جهاز الحاسب الخاص بك موجودا تحت هذا الفرع، وذلك لأنه يعمل كخادم محلي Local Server لتتمكن من خلاله من التعامل مع قواعد البيانات الموجودة على جهازك.

## الاتصال بقواعد بيانات Access:

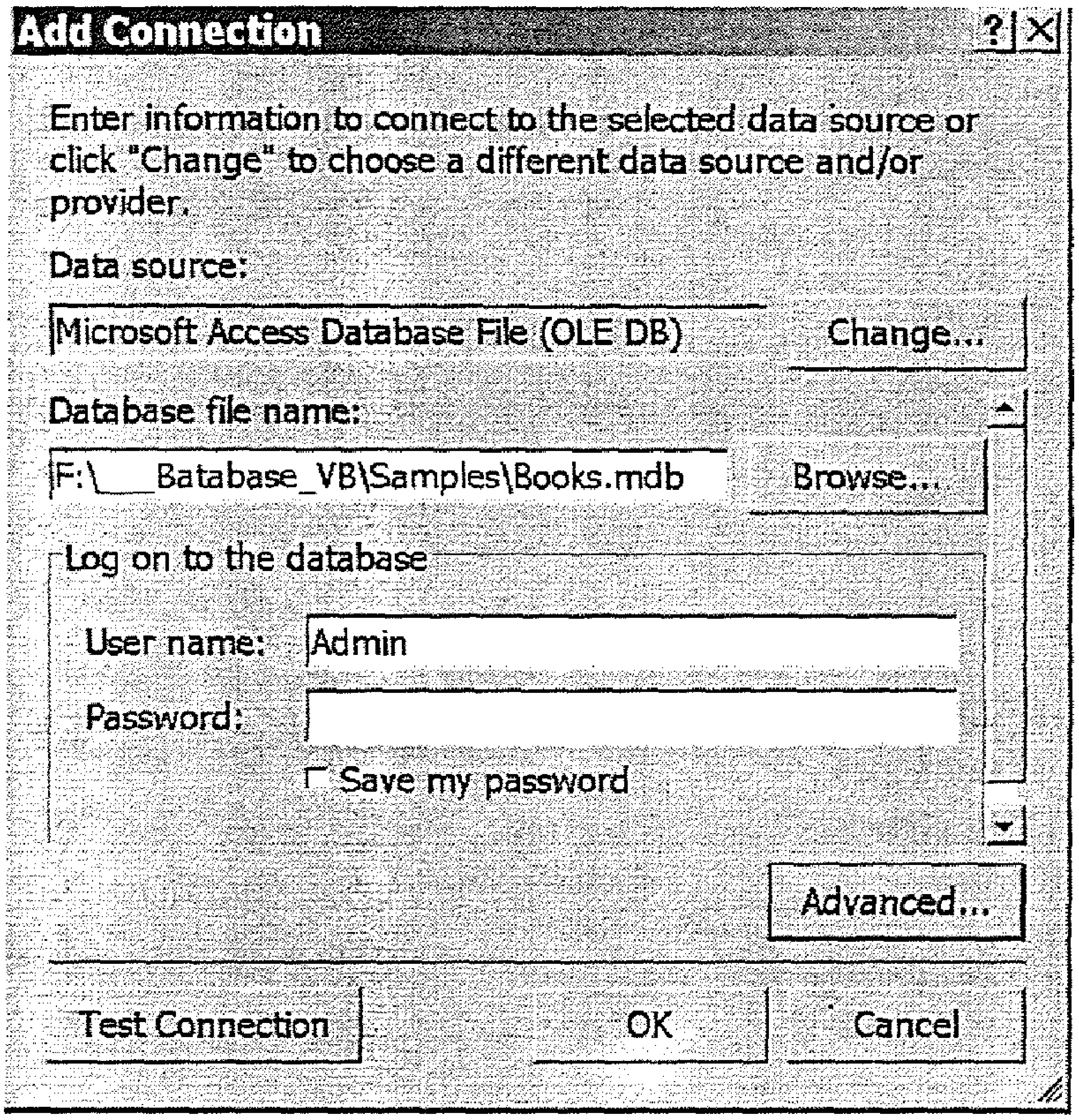
اضغط بزر الفأرة الأيمن العنصر Data Connections في متصفح الخـــوادم، ومن القائمة الموضعيّة اضغط الأمر "إضـــافة اتـــصال" Add Connection.. سيظهر لك مربّع حوار "اختيار مصدر البيانات" Choose data source، وفيـــه قائمة تحتوي على أسماء العديد من برامج قواعد البيانات:




اختر من القائمة العنصر: Microsoft Access Database File.. لاحظ وجود قائمة منسدلة أسفل النافذة، تتيح لك اختيار مزود البيانات Data Provider الذي ستستخدمه في برنامجك للاتصال بقاعدة البيانـــات.. بالنـــسبة لقواعـــد بيانـــات Access لن تحتوي القائمة المنسدلة إلا على المزود الخـــاص بقواعـــد بيانـــات OLE DB:

.Net Framework Data Provider for OLE DB

بعد هذا اضغط الزر Continue للانتقال إلى النافذة التالية، وهي نافذة "إضـــافة اتصال" Add Connection:



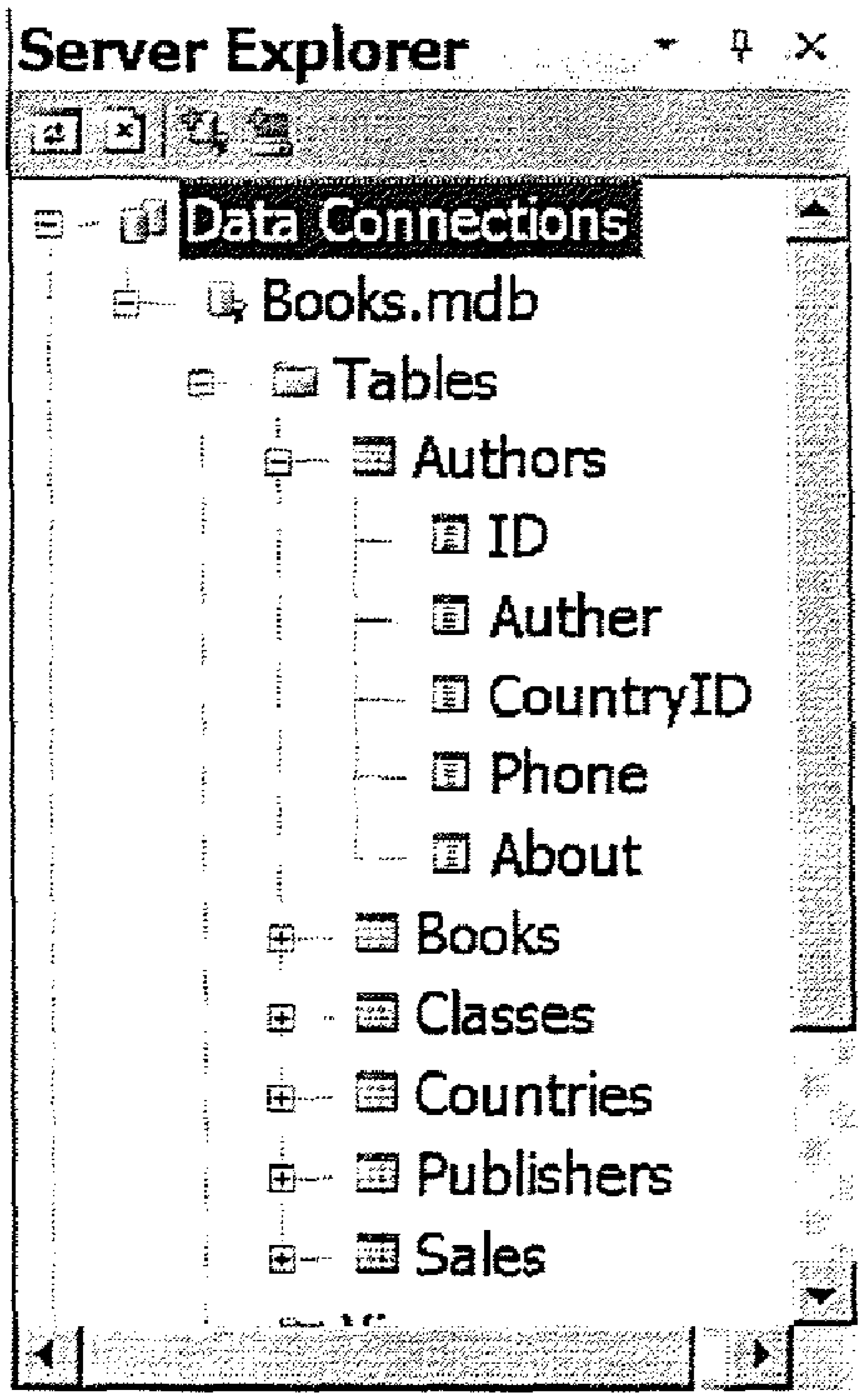

ستجد اسم مزود البيانات في الخانة العلوية في النافذة.. لو أردت تغيير هذا المزود، فيمكنك الضغط على الزر Change للرجوع إلى النافذة السابقة واختيار مزود لنوع قاعدة بيانات من نوع آخر.

الآن عليك اختيار ملف قاعدة البيانات، وذلك بكتابة مسار واسم الملف مباشرة في الخانة Database File Name، أو ضغط الزرّ Browse لاختيار ملفّ قاعدة البيانات من على القرص الصلب.. اخترِ الملفّ Books.mdb من الموضع الذي حفظته فيه على جهازك (ستجد هذا الملف على القرص الضوئي المرفق بهذا الكتاب).

ويتيح لك النصف السفلي من النافذة إدخال اسم المستخدم وكلمة السر الخاصة بقاعدة البيانات.. لكن نظرا لأننا لم نضع كلمة سر لقاعدة بيانات الكتب، فاترك هاتين الخانتين بدون تغيير: اسم المستخدم الافتراضيّ Admin، وكلمة السر فارغة.

ويمكنك ضغط الزر Test Connection لاختبار الاتصال بقاعدة البيانات.. ستظهر لك رسالة تؤكد نجاح الاتصال.. أما لو فشل الاتصال لسبب ما، فستظهر رسالة خطأ تخبرك أن هناك مشكلة.

اضغط الزر OK لإغلاق مربّع الحوار..
سيظهر اسم قاعدة البيانات Books.mdb في متصفح الخوادم، كما هو موضح في الصورة المجاورة.




اضغط العلامة "+" بجوار اسم قاعدة البيانات..
سيتم إسدال العناصر التي تتكون منها قاعدة البيانات.

أول عنصر من هذه العناصر هو الجداول Tables، ولو أسدلت ما تحتها من عناصر، فستجد أسماء الجداول التي أنشأناها في قاعدة البيانات.

ولو أسدلت عناصر أي جدول، فستجد تحته أسماء الأعمدة التي يحتويها.

-٣-

# قواعد بيانات SQL Server

كانت إصدارات فيجيوال ستديو السابقة لدوت نت، تدعم استخدام قواعد بيانات Access باعتبارها أهم قاعدة بيانات أصدرتها ميكروسوفت.. لكن مع ظهور دوت نت، احتلت قواعد بيانات SQL Server هذه المكانة، نظرا لأنها الأنسب للتعامل مع قواعد البيانات العملاقة التي يتصل بها آلاف المستخدمين في نفس اللحظة عبر شبكة داخلية Network أو عبر الشبكة الدولية Internet.. لعل هذا يوضح لك سبب وجود المصطلح "Server" في اسمها، فقواعد بيانات سيكويل سيرفر يجب أن تعمل على خادم محلي Local Server أو خادم موجود عبر الشبكة Online Server، ليقوم بتنظيم عمليات الاتصال بها وطلب البيانات منها.. ولو تم فصل قاعدة البيانات من هذا الخادم Disconnected، لا يمكن استخدامها إلا بعد إعادة توصيلها بالخادم مجددا Connected.. إذن فنحن نتعامل هنا مع خادم البيانات، ولا نتعامل مع ملف البيانات مباشرة كما كنا نفعل في قواعد بيانات آكسيس.

وهناك ثلاثة إصدارات شهيرة من سكيول سيرفر حتى الآن:

١- **SQL Server 2000:**

وهو متكامل مع دوت نت ٢٠٠٢ ودوت نت ٢٠٠٣، ولاستخدامه يجب أن تعد على جهازك أولا برنامجا اسمه:

Microsoft SQL Server Desktop Engine

أو اختصارا (MSDE)، والذي ستجده على أسطوانة Microsoft Office XP في المجلد MSDE2000.

٢- **SQL Server 2005:**

وهو متكامل مع دوت نت ٢٠٠٥ ودوت نت ٢٠٠٨، ولاستخدامه يجب أن تعد على جهازك أولا برنامجا اسمه:

SQL Server Management Studio Express

وهو نسخة مجانية خفيفة من SQL Server 2005 تمتلك قدرات أقل من النسخة الكاملة، لكنها كافية للغاية لإنشاء قواعد بيانات ضخمة يصل حجمها إلى ٤ جيجا بايت، ويمكنها أن تتصل بأكثر من ٣٢ ألف قاعدة

بيانات، والاستجابة لاتصال بضع مئات من المستخدمين في نفس اللحظة.. ويمكنك الحصول على نسخة من هذا البرنامج من صفحة تنزيل البرامج من موقع ميكروسوفت:

**http://www.microsoft.com/downloads**

في أعلى هذه الصفحة ستجد مربع البحث.. اكتب فيه النص:

SQL Server Management Studio

واضغط زر البحث (الذي تمثله أيقونة العدسة ).. ستجد في نتائج البحث رابط البرنامج الذي تبحث عنه.. اضغطه لعرض صفحة تنزيله، واضغط زر تنزيل النسخة المناسبة لنظام التشغيل الموجود على جهازك.

**٣- SQL Server 2008:**

وهو متكامل مع دوت نت ٢٠١٠، ولاستخدامه يجب أن تعد على جهازك أولا برنامجا اسمه:

SQL Server 2008 Management Studio Express

ويمكنك الحصول على نسخة منه من صفحة تنزيل البرامج من موقع ميكروسوفت بطريقة مماثلة للتي شرحنها سابقا.

## إعداد Management Studio Express:

قبل أن تبدأ إعداد هذا البرنامج على جهازك، عليّ أن أنبهك إلى ضرورة التأكد من أنك قد اخترت إعداد العنصر:

Microsoft SQL Server 2008 Express Edition

أثناء إعداد فيجيوال ستديو دوت نت ٢٠١٠، وإلا فإن عليك إعادة تشغيل ملف إعداد دوت نت، واختيار Add or Remove Features، ومن ثم وضع علامة الاختيار أمام هذا العنصر، حتى تستطيع فيجوال ستديو التعامل مع سيكويل سيرفر ٢٠٠٨.. أيضا، يحتاج برنامج الإعداد وجود البرامج التالية على جهازك:

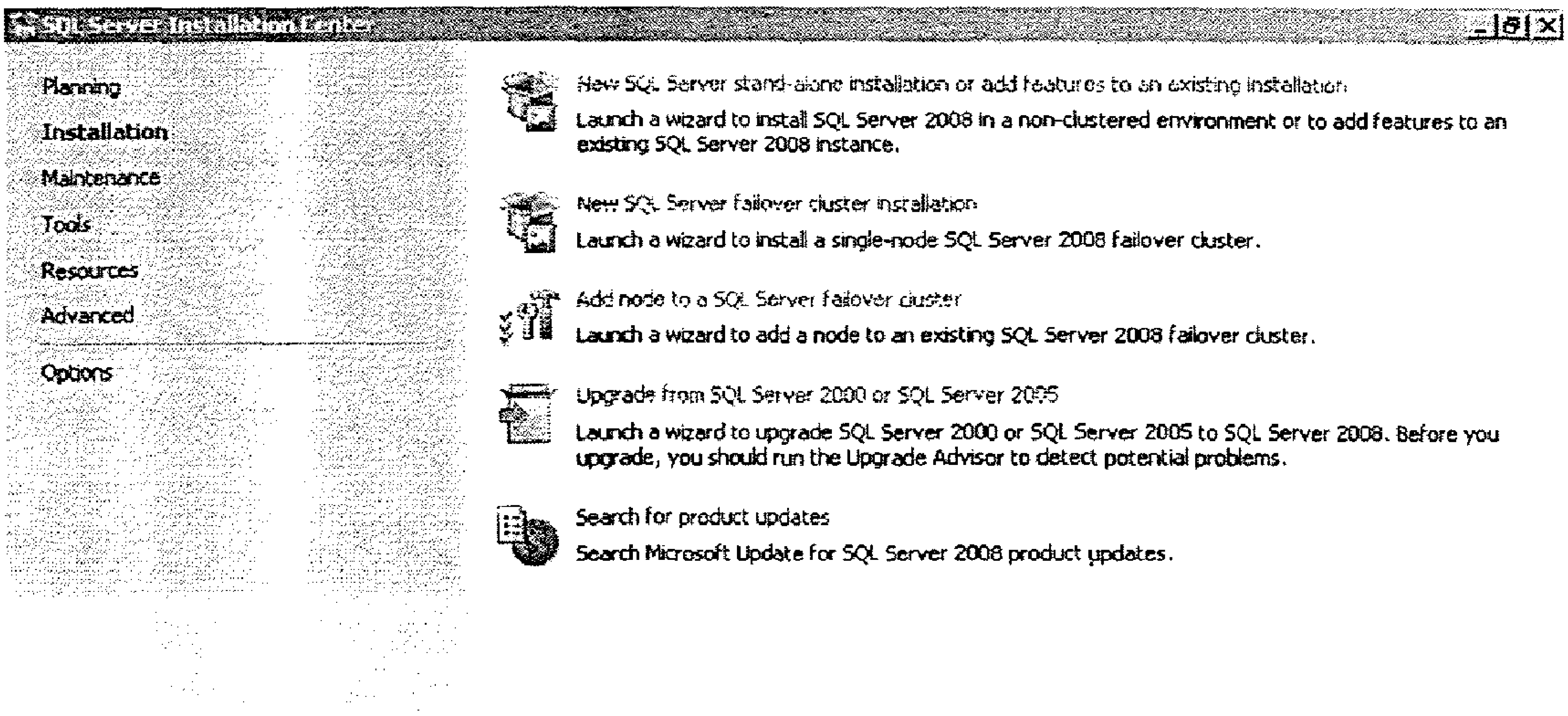
Windows Installer 4.5
WS-Management v1.1
Windows PowerShell V2 (CTP3)
.NET Framework 3.5 SP1

ويمكنك الحصول عليها من صفحة تنزيل البرامج من موقع ميكروسوفت كما شرحنا أعلاه.

والآن، اضغط برنامج الإعداد الخاص بـ:

SQL Server 2008 Management Studio Express

وانتظر حتى تظهر الشاشة الموضحة في الصورة:



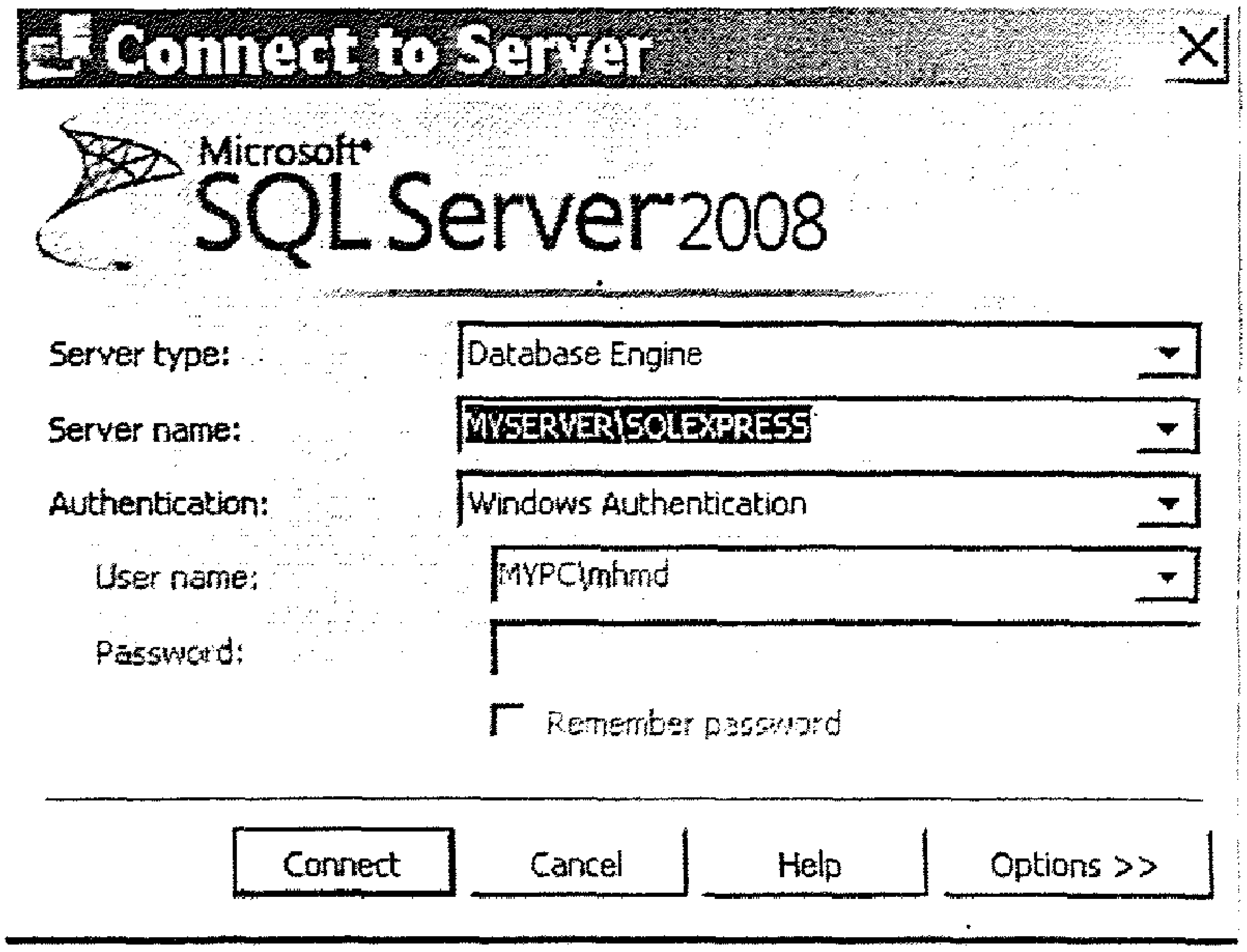

SQLServer2008

من الهامش الأيسر، اضغط الاختيار الثاني Install، حيث سيعرض لك الجزء الأيمن من النافذة عدة اختيارات.. إذا لم تكن قد أعددت إصدارا سابقا من سيكويل سيرفر على جهازك، فاضغط الاختيار الأول.. سيتم أولا التأكد من وجود المتطلبات اللازمة للإعداد على جهازك.. فإن كانت كلها موجودة، فاضغط OK وواصل باقي خيارات الإعداد.. وتأكد دائما من ضغط الزر Select All في كل صفحة تعرض العناصر المتاحة للإعداد، لتضمن إعداد كافة مكونات البرنامج.

بعد انتهاء عملية الإعداد، توجه إلى سطح المكتب Desktop واضغط:

Start\Programs\Microsoft SQL Server 2008\SQL Server Management Studio

سيؤدي هذا إلى فتح مدير قواعد بيانات سيكويل سيرفر SSMS، وأول ما سيعرضه لك هو نافذة الاتصال بخادم قاعدة البيانات:



هذه النافذة تتيح لك تحديد ثلاث خيارات:

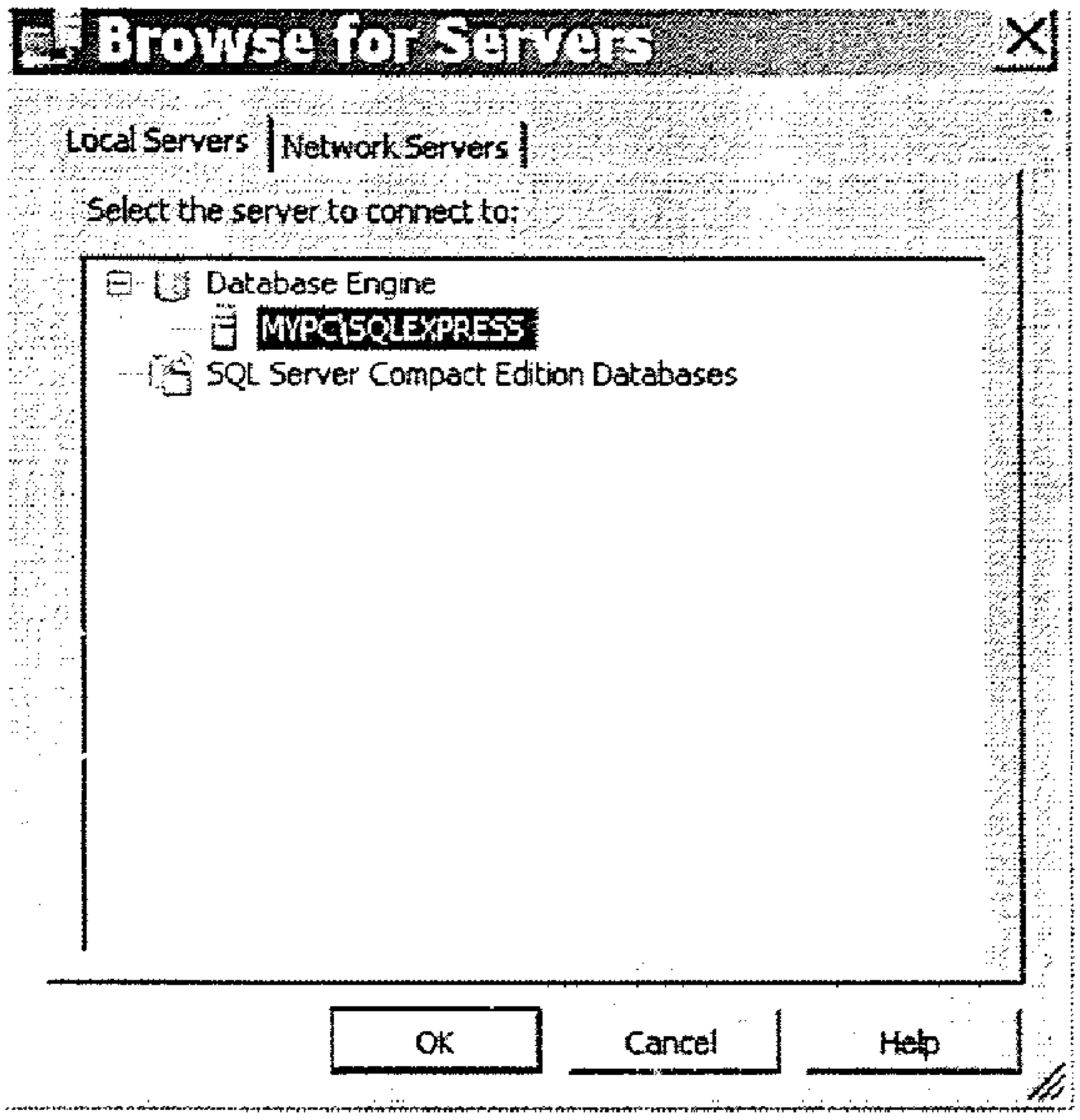
**١- نوع محرك خادم البيانات Server Type:**

تعرض لك هذه القائمة المنسدلة نوعين من الخوادم:

أ- Database Engine: وهو النوع الذي سنستخدمه في برامج الويندوز وصفحات المواقع ASP.NET.

ب- SQL Server Compact Edition: وهو النوع المستخدم مع الأجهزة الكفية المحمولة.

**٢- اسم الخادم Server Name:**




يمكنك كتابة اسم خادم البيانات في هذه القائمة مباشرة، أو ضغط زر الإسدال لاختيار اسم الخادم، حيث ستعرض لك القائمة المنسدلة أسماء الخوادم التي اخترتها سابقا، بالإضافة إلى العنصر <Browse for more...>، ولو ضغطت هذا العنصر فستظهر لك نافذة اختيار الخادم، وهي تعرض شريطين علويين 2 Tabs، أولهما يتيح لك اختيار خادم محلي (موجود على جهازك)، والآخر يتيح لك اختيار خادم موجود على الشبكة Network.. وسنستخدم في هذا الكتاب خادما محليا Local Server، ولو ضغطت العلامة + الموجودة بجوار Database Engine، فستجد تحتها أسماء الخوادم المحلية الموجودة على جهازك.. وعلى الأقل ستجد خادما واحدا، هو خادم سيكويل سيرفر، حيث يتكون اسم هذا الخادم من اسم جهازك (وليكن MyPC) يليه اسم خادم سيكويل سيرفر SQLEXPRESS.. اختر هذا الخادم واضغط OK للعودة إلى النافذة الأولى.. ستجد في مربع النص اسم الخادم MYPC\SQLEXPRESS.. لاحظ مجددا أننا سنستخدم في هذا الكتاب الاسم MyPC باعتباره اسم الجهاز.. لكن لو كان اسم جهازك مثلا MohammadPC فستجد في هذه الخانة الاسم: MohammadPC\SQLEXPRESS... وهكذا.

### ٣-تحقيق الهوية Authentication:

تتيح لك هذه القائمة المنسدلة نوعين من تحقيق الهوية:

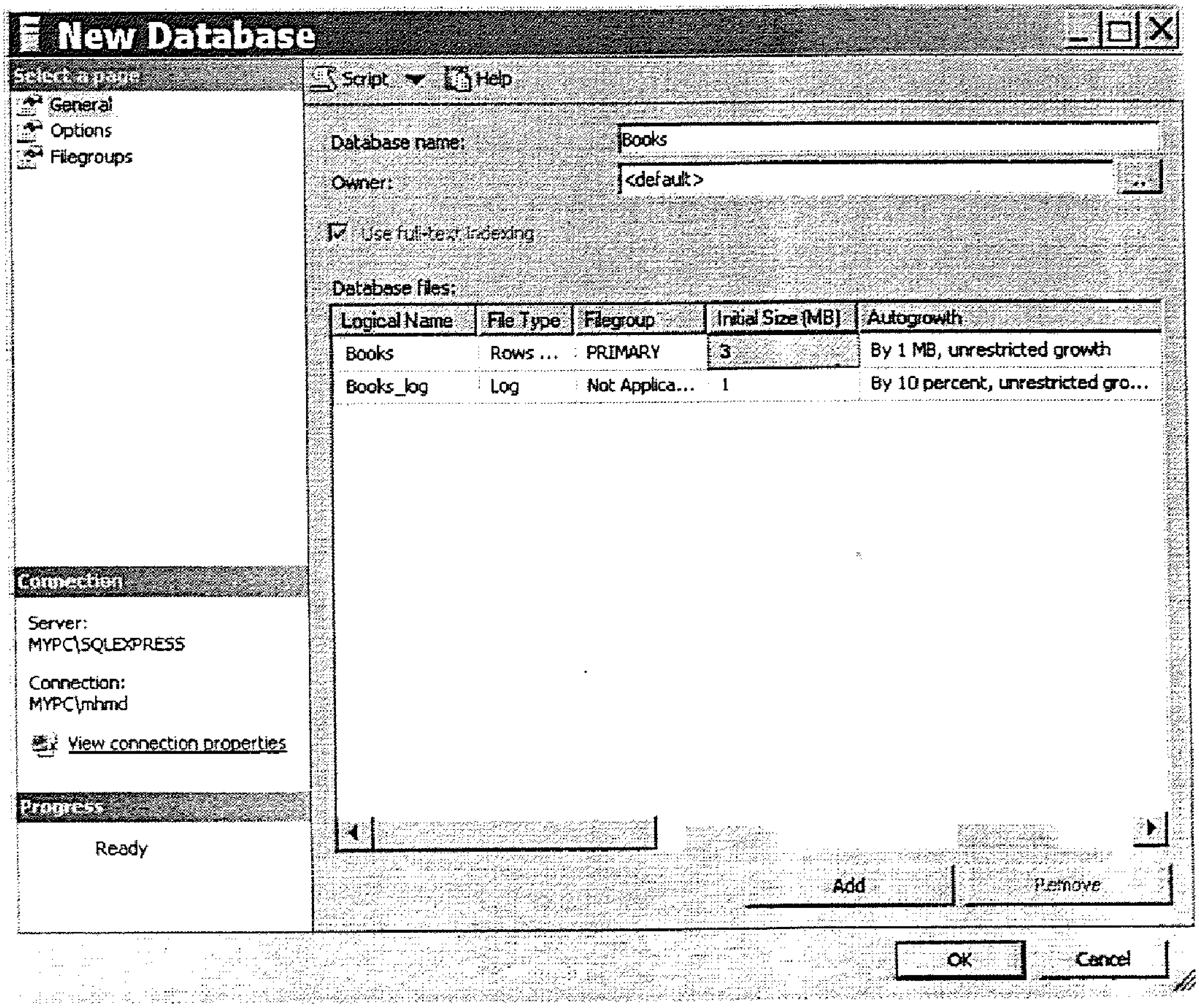
أ- تحقيق الهوية بواسطة الويندوز Windows Authentication:
في هذا النوع لا يتم حماية الاتصال بالخادم باسم مستخدم وكلمة سر خاصة، بل تكون الحماية معتمدة على بيانات مستخدم الويندوز.. بمعنى أن مستخدم الويندوز الذي أنشأ الاتصال يستطيع الاتصال بالخادم والتعامل مع قواعد البيانات المتاحة بمجرد تسجيل دخوله على الويندوز.. وهذا هو نوع تحقيق الهوية الذي سنستخدمه مع الخادم المحلي لأنه غير محمي.

ب- تحقيق هوية خادم سيكويل SQL Server Authentication:
في هذا النوع تتم حماية الاتصال بالخادم وكل قواعد البيانات التي ستنشئها عليه باسم مستخدم Username وكلمة مرور Password، حيث يتم تفعيل مربعي النص الخاصين باسم المستخدم وكلمة المرور.. ويستخدم هذا النوع مع الخوادم الموجودة على الشبكة، لأنها تكون محمية ضد محاولات الاختراق، ويكون لكل مستخدم صلاحيات معينة في التعامل مع خادم البيانات كما يحددها مدير النظام.. فهناك مستخدمون يستطيعون إضافة أو حذف قواعد البيانات أو الجداول والأعمدة والسجلات، بينما تقتصر صلاحيات بعض المستخدمين على قراءة البيانات من بعض الجداول.. لهذا في الغالب لا تسمح الخوادم الحقيقية (غير الخادم المحلي الافتراضي) بالاتصال بها بدون أن يحدد المتصل اسمه وكلمة مروره.

بعد أن تنتهي من إدخال هذه البيانات، اضغط OK لإغلاق النافذة وإنشاء الاتصال.. سيؤدي هذا إلى إضافة اسم الخادم الذي اتصلت به إلى متصفح الكائنات Object Explorer الموجود على يسار نافذة مدير سيكويل سيرفر، ولو أسدلت العناصر الفرعية لهذا الاتصال، فستجد من ضمنها العنصر Databases، وهو العنصر الذي تظهر تحته كل قواعد البيانات التي تعمل على هذا الخادم، كما يمكنك من خلال هذا العنصر إضافة قواعد بيانات جديدة إلى الخادم، أو تشغيل قواعد بيانات موجودة مسبقا.

## إنشاء قاعدة بيانات سيكويل سيرفر:

في متصفح الكائنـــات Object Explorer، اضـــغط فـــرع قواعـــد البيانـــات Databases بزر الفأرة الأيمن، ومـــن القائمـــة الموضـــعية Context Menu اضغط الأمر New Database لعرض نافذة إنشاء قاعدة بيانات جديدة.




هذه النافذة تتيح لك إنشاء أكثر من قاعدة بيانات، حيث تحتوي على جدول يعرض تفاصيل ملفات قواعد البيانات التي تريد إنشاءها.. كل ما عليك هو كتابـــة اســـم قاعدة البيانات في الخانة Database Name أعلى النافذة، حيث ستلاحظ ظهور ملفين في الجدول السفلي يحملان اسم هذه القاعدة.. فمـــثلا، لـــو كتبـــت الاســـم Books في خانة اسم قاعدة البيانات، فسيعرض الجدول الملفين الآتيين:

**١- Books.mdf:**

هذا هو الملف الذي يحفظ بيانات قاعدة البيانات، وستجد أن خانة النـــوع المجاورة لاسمه تشير إلى أنه Rows Data أي أنـــه يحمـــل بيانـــات السجلات.. وستجد أن خانة الحجم المبدئي تشير إلى أن حجم هذا الملف سيكون مبدئيا ٣ ميجا، وسيزيد بمعدل ١ ميجا في كل مرة تتجاوز فيهـــا

البيانات حجم الملف.. ويمكنك أن تغير الحجم المبدئي ومعدل الزيادة تبعا لنوع قاعدة البيانات التي تتعامل معها، فلو كنت تتوقع معدلا كبيرا لإضافة البيانات إليها، فالأفضل أن تجعل معدل زيادة حجمها أكبر، لأن هذا يوفر الوقت المستهلك عند تكرار عملية تكبير الملف.

**٢- Books_log.ldf:**

هذا هو الملف الذي يحفظ سجل أداء قاعدة البيانات Log، وستجد أن خانة النوع المجاورة لاسمه تشير إلى أنه Log.. وستجد أن خانة الحجم المبدئي تشير إلي أن حجم هذا الملف سيكون مبدئيا ١ ميجا، وسيزيد بمعدل ١٠% من حجمه في كل مرة تتجاوز فيها البيانات حجم الملف.. ويمكنك أن تغير الحجم المبدئي ومعدل الزيادة تبعا لحاجتك.

لاحظ أن ملف سجل الأداء Log يخزن كل التفاصيل التي تتم على قاعدة البيانات من إضافة وحذف واستعلامات وغيرها، ومن قام بها ومتى، وهذا يجعل حجمه يتضخم بسرعة كبيرة، لأنه سيحتوي على البيانات الأصلية وكل التعديلات التي تمت عليها.. هذا يحقق الفوائد الآتية:

١- يقدم تاريخا مفصلا للعمليات التي تمت على قاعدة البيانات، ويسهل مراقبة أدائها ومعرفة حجم نشاط المستخدمين المتعاملين معها.

٢- يتيح لك استعادة أي وضع سابق لقاعدة البيانات، ليمكنك التراجع عن أية عملية غير مرغوبة لاستعادة حالة قاعدة البيانات قبل إجرائها.

٣- يحمي البيانات من الضياع.. ففي قاعدة بيانات Access يمكن أن يؤدي انقطاع الكهرباء أو حدوث أي مشكلة في الويندوز أثناء حفظ بعض البيانات إلى تدمير ملف قاعدة البيانات كلها، لأن عملية الحفظ تتم في الملف مباشرة.. بينما في سيكويل سيرفر يتم حفظ التغييرات في ملف سجل الأداء Log أولا، وبعد مدة زمنية معينة يتم نسخ كل التغييرات من ملف سجل الأداء إلى ملف قاعدة البيانات.. هذا يعني أنه لو حدثت أي مشكلة أثناء الحفظ في سجل الأداء فإن ملف قاعدة البيانات سيظل سليما، وستكون البيانات المفقودة أقل ما يمكن، حيث سيتم فقد الجزء الذي كان يتم حفظه فقط.. أما لو حدثت مشكلة أثناء نقل البيانات من ملف سجل الأداء إلى ملف قاعدة البيانات وأدت إلى تدمير ملف قاعدة البيانات، فلن تفقد أي بيانات، لأن سجل الأداء ما زال يحتفظ بنسخة منها.

وتعرض الخانة Path مسار حفظ كل ملف من هذين الملفين، وهو مبدئيا:

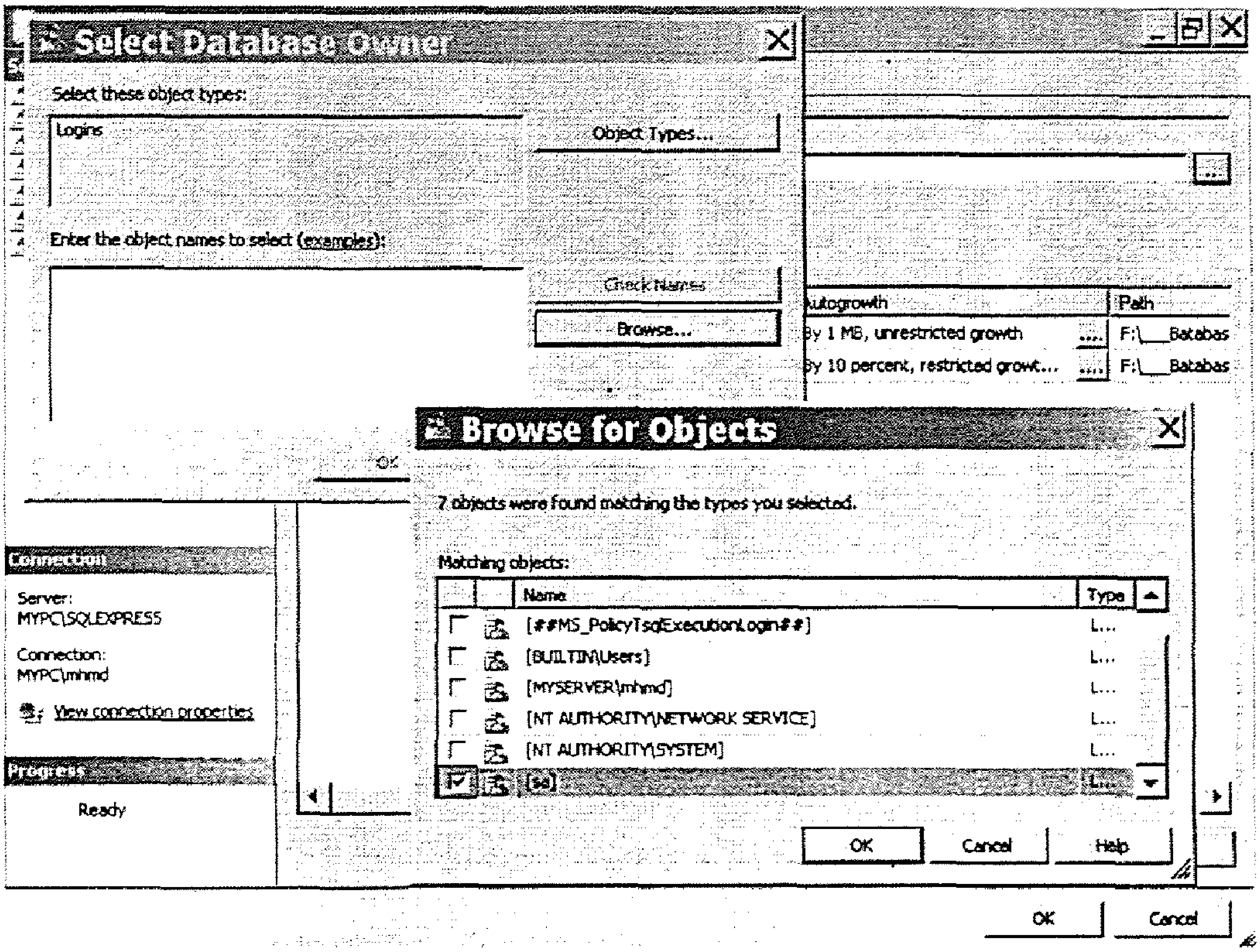
C:\Program Files\Microsoft SQL Server\MSSQL10.SQLEXPRESS\MSSQL\DATA

ويمكنك تغيير هذا المسار بضغط زر الانتقال الموجود في هذه الخانة، لعرض مربع حوار اختيار مجلد.

| **تنبيه:** |
| --- |
| أنصحك بتغيير هذا المسار إلى أي مسار آخر خاص بك على جزء من القرص الصلب غير الذي يوجد عليه نظام الويندوز، حتى لا تفقد قواعد بياناتك في عملية تهيئة Format طائشة، أو بسبب حدوث أي تدمير لنظام التشغيل لأي سبب كان، يجعلك تعيد إعداده دون أن تنتبه إلى أخذ نسخة من ملفاتك الهامة. |

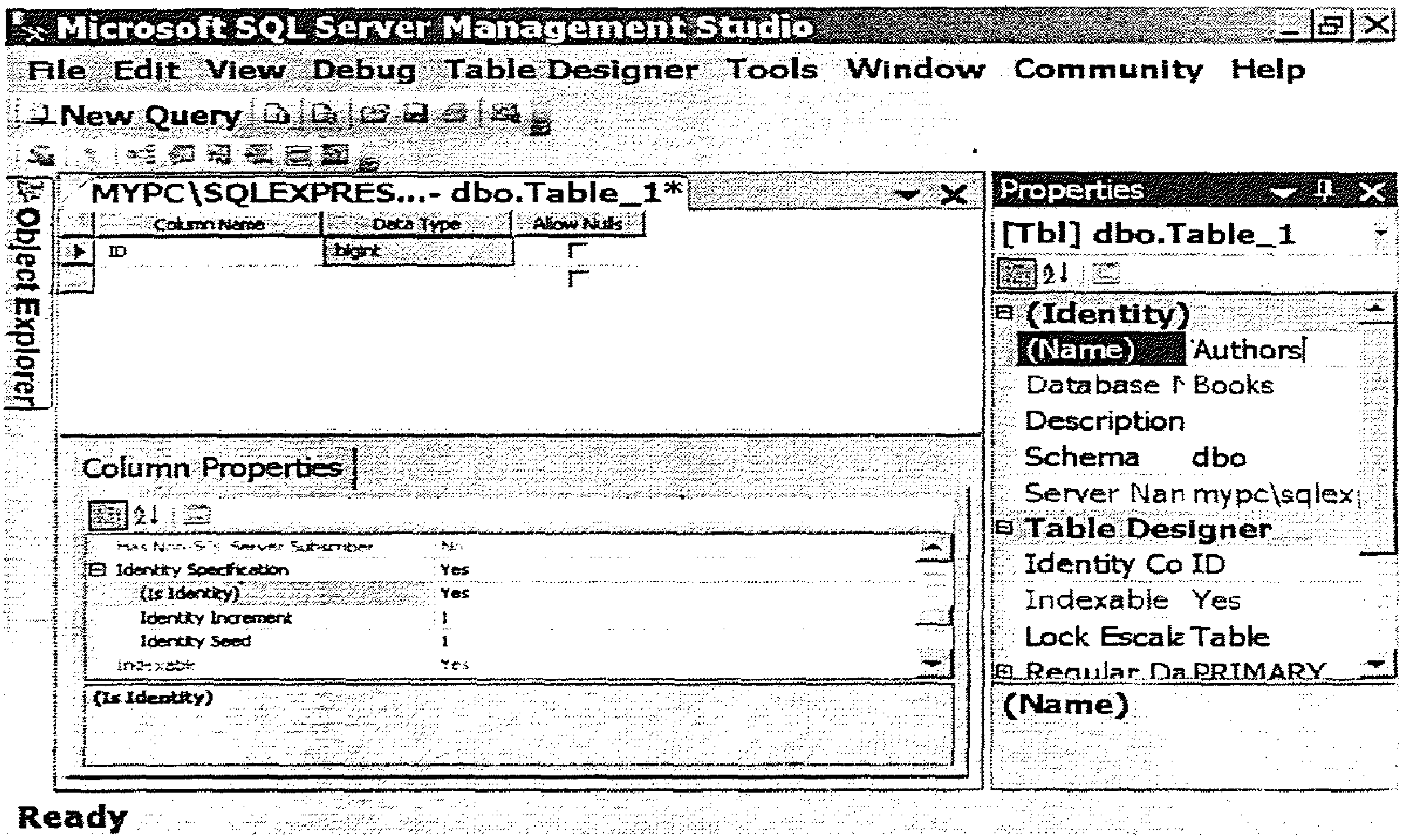
ولإنشاء قاعدة بيانات أخرى، اضغط الزر Add الموجود أسفل النافذة، حيث سيظهر صفان جديدان في الجدول.. حدد أي صف منهما، واكتب اسم قاعدة البيانات الجديدة في الخانة Database Name أعلى النافذة.. وإذا أردت حذف أي قاعدة بيانات، فحدد أيا من ملفيها، واضغط الزر Remove أسفل النافذة.

يتبقى جزء مهم في هذه النافذة، هو خانة مالك قاعدة البيانات Owner.. لو تركت قيمة هذه الخانة <Default> فسيكون اسم المستخدم الخاص بك على الويندوز Username هو مالك هذه القاعدة، لكن هذا قد يحرمك بعض الصلاحيات.. لهذا لو أردت صلاحيات كاملة، فاضغط الزر المجاور لهذه الخانة لعرض مربع حوار اختيار مالك قاعدة البيانات.. اضغط الزر Browse الموجود في النصف السفلي من هذه النافذة، ومن النافذة التي ستظهر اختر [sa] وهو اختصار لتعبير مدير النظام System Administrator واضغط OK لإغلاق النافذتين.



بعد أن تفرغ من إنشاء كل قواعد البيانات التي تريدها، اضغط الزر OK لإغلاق النافذة وإنشاء الملفات.. غني عن الذكر أن ضغط الزر Cancel سيلغي كل ما فعلته ولن يتم إنشاء أي قاعدة من التي أدخلت أسماءها، فكل ما تدخله في هذه النافذة هو مجرد تفاصيل، ولا تدخل حيز التنفيذ إلا بعد ضغط الزر OK.

الآن، ستجد اسم قاعدة البيانات Books تحت الفرع Databases في متصفح الكائنات.. اضغط العلامة + المجاورة لهذا الاسم، لإسدال عناصر قاعدة البيانات.. ستجد ضمن هذه العناصر العنصر Tables، وهو العنصر الذي يحتوي على كل جداول قاعدة البيانات.. اضغط هذا العنصر بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر New Table لإنشاء جدول جديد.. سيؤدي هذا إلى عرض نافذة تصميم الجدول في مدير سيكويل، كما ستظهر نافذة الخصائص التي تعرض خصائص هذا الجدول.. الجميل في الأمر هو أن بيئة مدير سيكويل تشبه إلى حد بعيد بيئة التطوير المتكاملة IDE الخاصة بدوت نت، مما سيشعرك بالألفة معها، ويجعل استخدامك لها سهلا وسريعا.



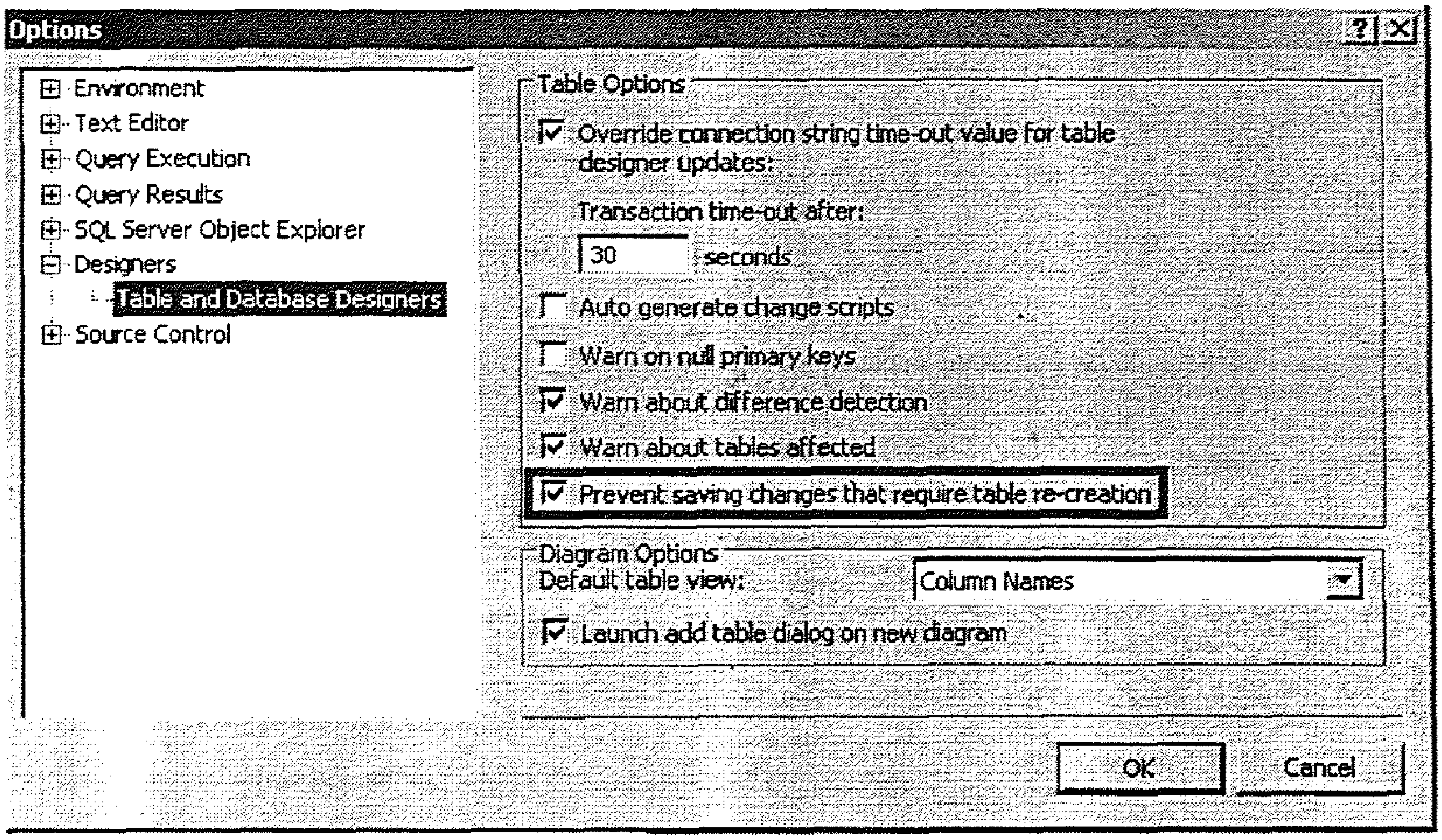

دعنا ننشئ جدول المؤلفين:

أدخل أسماء الأعمدة ونوع كل منها في الجدول العلوي بنفس الطريقة التي استخدمناها في آكسيس، واستخدم الجزء السفلي من النافذة للتحكم في خصائص كل نوع.. وإذا لم تكن تريد السماح للمستخدم بترك أي خانة من خانات هذا العمود فارغة، فأزل علامة الاختيار من مربع الاختيار Allow Nulls.. ولا تنسَ تغيير اسم الجدول من Table1 إلى Authors، ثم اضغط زر الحفظ من شريط الأدوات.

**ملحوظة:**

بعد أن تقوم بحفظ تصميم الجدول قد يرفض سيكويل سيرفر إجراء أي تعديل على هذا التصميم (كإضافة أو حذف أعمدة)، مع ظهور رسالة تخبرك أن التغييرات التي قمت بها تحتاج إلى إنشاء الجدول من جديد!

لحل هذه المشكلة افتح SQL Server Management Studio، ومن القائمة Tools اضغط الأمر Options، وفي نافذة الخيارات أسدل العنصر Designers من القائمة اليسرى، واضغط العنصر الفرعي Table and Database Designers كما هو موضح في الصورة:




وفي الجزء الأيمن، تأكد من إزالة علامة الاختيار من الخيار:

Prevent Saving Changes that require table re-creation.

فهذا الخيار يمنع حفظ التغييرات التي تسبب إعادة إنشاء الجدول.. اضغط Ok لحفظ التغييرات وإغلاق النافذة.

أما إن استمر هذا الخطأ في الحدوث في بيئة دوت نت، فاستخدم خادم سيكويل لإجراء التعديل الذي تريده في تركيب الجدول.. فإذا أعيتك الحيل، فبإمكانك نسخ تصميم جميع الأعمدة ولصقها في جدول جديد.. لفعل هذا، افتح نافذة تصميم الجدول واضغط هامش العمود الأول بالفأرة، ثم اضغط الزر Shift من لوحة المفاتيح، واضغط هامش العمود الأخير بالفأرة، وبهذا سيتم تحديد كل الأعمدة.. اضغط Ctrl+C لنسخ الأعمدة، ثم أنشئ جدولا جديدا، وفي نافذة التصميم اضغط هامش العمود الفارغ بالفأرة، واضغط Ctrl+V للصق الأعمدة التي نسختها، ثم أجر عليها التعديلات التي تريدها، أو أضف إليها أعمدة جديدة.. بعد هذا احذف الجدول القديم، وسمِّ الجدول الجديد باسم الجدول المحذوف.

لو أغلقت النافذة الآن، فسترى اسم جدول المؤلفين أسفل العنصر Tables، وسيكون كالتالي dbo.Authors، حيث إن الحروف dbo هي اختصار للتعبير "مالك قاعدة البيانات" Database Owner، وهو مستخدم افتراضي لا يمكن حذفه.. والسبب في إضافة هذا الجدول إلى المستخدم dbo هو أننا تركنا للخاصية Schema الخاصة بالجدول القيمة dbo (انظر نافذة الخصائص)، ولو غيرناها إلى Guest مثلا، فسيصير اسم الجدول Guest.Authors، لأنه سينتمي إلى المستخدم Guest.. وفي حالات أخرى، يضاف الجدول إلى اسم المستخدم الذي سجل دخوله على قاعدة البيانات، وذلك إذا لم يكن يملك صلاحيات مدير قاعدة البيانات Administrator.

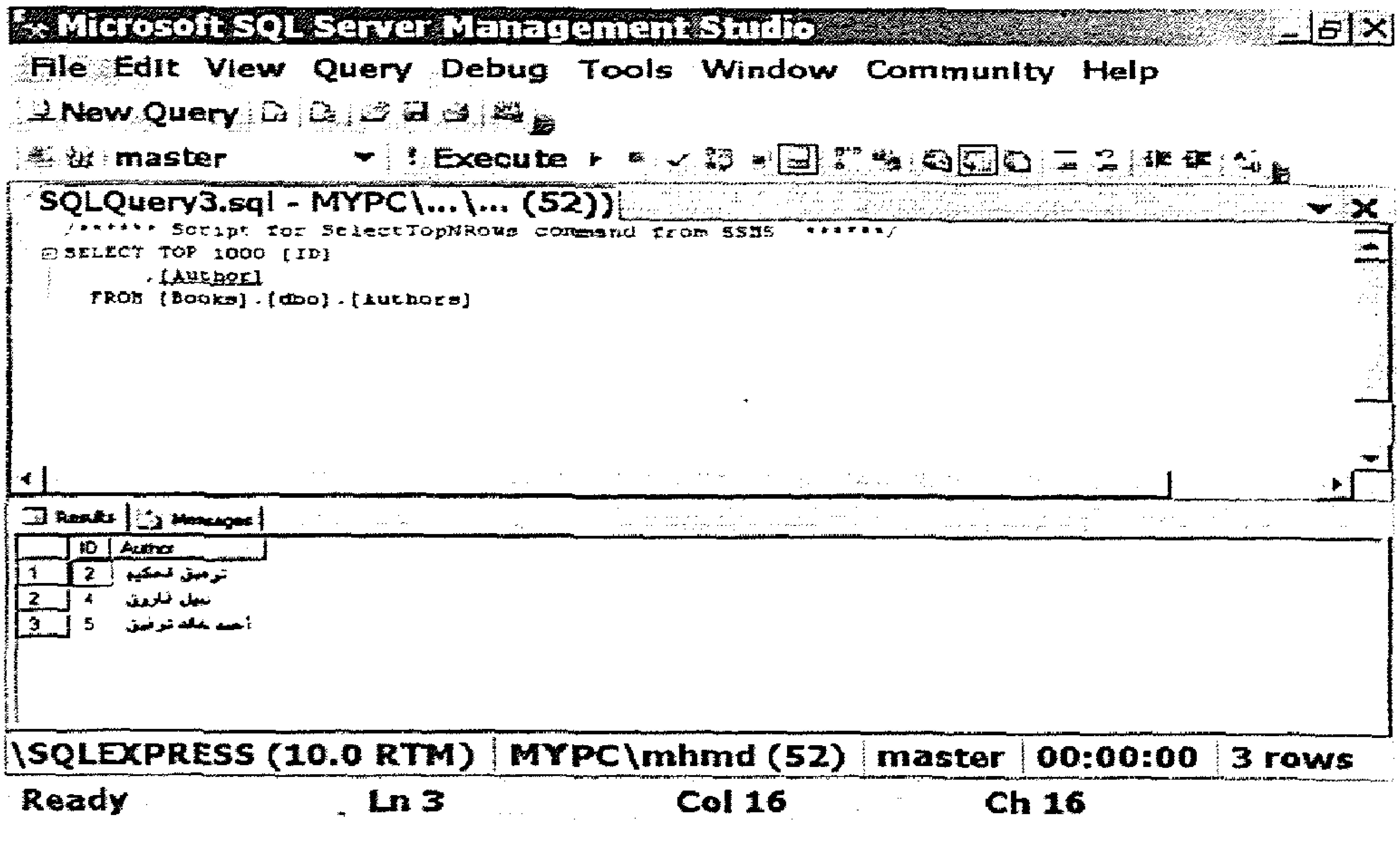
والآن، اضغط العنصر dbo.Authors بزر الفأرة الأيمن.. ستعرض لك القائمة الموضعية مجموعة من الأوامر الهامة، منها:

- **Design:**

يعرض نافذة تصميم أعمدة الجدول.

- **Select Top 1000 rows:**

يعرض أول ١٠٠٠ سجل في الجدول، وفي هذه الحالة لن تتمكن من تعديل أي شيء في هذه السجلات.. لاحظ أن الرقم ١٠٠٠ يهدف إلى تقليل الوقت المستهلك في عرض الجداول الضخمة، لكن لو شئت أن تعرض أي عدد تريده من السجلات وبأي شرط، فإن الجزء العلوي من النافذة يعرض جملة الاستعلام التي تم استخدامها لعرض الألف سجل، ومن ثم يمكنك تعديلها كما تشاء، ثم تضغط زر الفأرة الأيمن، ومن القائمة الموضعية تضغط الأمر Execute لتنفيذ الاستعلام وعرض ناتجه في الجدول السفلي.. وسنتعلم في فصل لاحق كيف نتعامل مع جمل الاستعلام SQL Queries.



- **Edit Top 200 Rows:**

يؤدي هذا الأمر إلى عرض نافذة تحرير السجلات، وبها أول ٢٠٠ سجل إن وجدت، حيث يمكنك تعديل بيانات السجلات الموجودة أو حذفها، كما يمكنك إضافة سجلات جديدة.

- **Rename:**

يؤدي إلى تحول اللافتة التي تعرض اسم الجدول إلى مربع نص، ليمكنك تغيير اسمه.. لاحظ أنك تستطيع فعل نفس الشيء مباشرة، بمجرد ضغط لافتة اسم الجدول ضغطة واحدة بالفأرة بعد تحديده، تماما كما تفعل مع أسماء الملفات في متصفح الويندوز.

- **Delete:**

يؤدي إلى حذف الجدول من قاعدة البيانات، حيث ستظهر لك شاشة تعرض اسم الجدول لتأكيد أمر حذفه، فإن كنت مصرا فاضغط OK، وإن رجعت عن عزمك فاضغط Cancel.. وطبعا يمكنك فعل نفس الشيء بتحديد اسم الجدول وضغط الزر Delete من لوحة المفاتيح مباشرة.

- **Properties:**

تعرض نافذة بها خصائص الجدول والكثير من تفاصيله.

والآن، بعد أن انتهينا من جدول المؤلفين، يمكنك اتباع نفس الطريقة لإنشاء باقي جداول قاعدة الكتب، لتكون كتلك التي أنشأناها في آكسيس.. وستجد قاعدة بيانات الكتب Books.mdf على القرص الضوئي المرفق بهذا الكتاب.

## أنواع البيانات في سيكويل سيرفر:

الجدول التالي يلخص لك أهم أنواع البيانات الخاصة بالأعمدة التي تنشئها في سيكويل سيرفر:

| النوع | الحجم (Bytes) | الشرح |
|---|---|---|
| **bigint** | ٨ | عدد صحيح كبير، وهو يقبل أرقاما من -٢^٦٣ إلى ٢^٦٣ -١ (وهو رقم ضخم جدا يساوي تقريبا ١٠ مليار مليار). |
| **int** | ٤ | عدد صحيح، وهو يقبل أرقاما من -٢^٣١ إلى ٢^٣١ -١ (وهو رقم يساوي تقريبا ٢ مليار). |
| **smallint** | ٢ | عدد صحيح صغير، وهو يقبل أرقاما من -٢^١٥ إلى ٢^١٥ -١ (وهو رقم يساوي ٣٢،٧٦٧). |
| **tinyint** | ١ | وحدة ثنائية، وهي تقبل أرقاما من ٠ إلى ٢٥٥. |
| **bit** | | خانة ثنائية واحدة، تقبل صفرا أو ١، وهي مفيدة عند التعامل مع قيم منطقية (Yes أم No) أو (True أم False). |
| **decimal(x, y)** أو **numeric(x, y)** | ١٧ | عدد عشري، يتيح لك تحديد الدقة العشريّة التي تريد التعامل معها، حيث x تمثل عدد أرقام العدد (بحد أقصى ٣٨ رقما)، و y تمثل عدد الخانات العشرية في هذا العدد. |
| **money** | ٨ | قيمة نقدية، تتيح لك التعامل مع كسور تصل دقتها العشرية إلى ٤ خانات عشرية، وتنحصر القيم المقبولة بين -٩٢٢٣٣٧٢٠٣٦٨٥،٤٧٧,٥٨٠٨ و +٩٢٢٣٣٧٢٠٣٦٨٥٤٧٧,٥٨٠٧. |
| | ٤ | قيمة نقدية، تتيح لك إدخال كسور تصل دقتها العشرية إلى ٤ خانات عشرية، وتنحصر القيم المقبولة بين -٢١٤٧٤٨,٣٦٤٨ و +٢١٤٧٤٨,٣٦٤٧. |

| | | |
|---|---|---|
| **float** | ٤ أو ٨ | عدد عشري مقرب، تتحدد مساحة تخزينه تبعا لعدد الخانات العشرية المطلوبة.. فلو اخترت أن تكون دقته العشرية ٧ خانات، فسيتم حفظه في ٤ وحدات 4 Bytes، أما لو اخترت أن تكون دقته العشرية ١٥ خانة، فسيتم حفظه في ٨ وحدات 8 Bytes. |
| **real** | ٤ | عدد حقيقي، وهو حالة خاصة من النوع float، فأقصى دقة عشرية له هي ٧ خانات. |
| **datetime** | ٨ | تاريخ ووقت تصل دقته إلى الميلي ثانية ممثلة في ٣ خانات عشرية.. مثال:<br>'01/01/1998 12:59:59.999'<br>ولا يقبل هذا النوع إلا التواريخ التي تقع بين ١٧٥٣/٠١/٠١ و ٩٩٩٩/١٢/٣١. |
| **datetime2(n)** | من ٦ إلى ٨ | تطوير للنوع datetime، يتيح لك التعامل مع تواريخ من ١/١/١ إلى ٩٩٩٩/١٢/٣١.. ويمثل الرقم n عدد الخانات العشرية التي تريد استخدامها مع أجزاء الثانية.. فمثلا لو أردت التعامل مع أجزاء من مئة ألف جزء من الثانية، فاجعل قيمة n تساوي ٥. |
| **smalldatetime** | ٤ | تاريخ ووقت تصل دقته إلى الدقيقة فقط.. مثل: 01/01/1998 12:30 |
| **date** | ٣ | تاريخ فقط بدون أي وقت، مثل:<br>01/01/1998 |
| **time(n)** | من ٣ إلى ٥ | وقت بدون تاريخ.. ويمثل الرقم n عدد الخانات العشرية المستخدمة لعرض أجزاء الثانية. |
| **datetimeoffset(n)** | من ٨ إلى ١٠ | تاريخ ووقت مرفق به فارق التوقيت عن التوقيت العالمي.. مثال:<br>01/01/1998 12:35:29.123 +2:15<br>هذا المثال يوضح أن هذا التاريخ مرتبط بموقع جغرافي يزيد بساعتين وربع عن توقيت خط جرينتش.<br>ويمثل الرقم n عدد الخانات العشرية التي تريد استخدامها مع أجزاء الثانية. |

| | | |
|---|---|---|
| نص ثابت الطول يتكون من عدد n من الحروف.. لاحظ أن هذا النوع لا يدعم الكود الموسع Unicode، لهذا ستكون الحروف المسموح بكتابتها هي فقط الحروف المتاحة على الخادم Server الذي توجد عليه قاعدة البيانات.. انتبه لهذا جيدا، لأنك في المشاريع الحقيقية قد تضع قاعدة البيانات على خادم في دولة أجنبية، وبالتالي لن يدعم اللغة العربية.<br>ويمكنك التحكم في عدد الحروف، بتغيير قيمة الخاصية Length من الجزء السفلي من نافذة تصميم الجدول، مع ملاحظة أن الحد الأقصى لطول هذا النص هو ٨٠٠٠ حرف.. والقيمة الافتراضية لطول النص هي ١٠ حروف، وفي هذه الحالة لو حاولت كتابة نص طوله ١١ حرفا فسيرفض سيكويل سيرفر ذلك، بينما لو كتبت نصا طوله ٨ حروف فسيتم إكماله إلى ١٠ حروف بإضافة مسافتين في نهايته. | n | **char(n)** |
| مشابه للنوع char في كل شيء إلا في أمر واحد، وهو أنه يمثل نصا مرنا متغير الطول لا يقبل الحروف الموسعة، وأقصى عدد يقبله من الحروف هو n (الحد الأقصى لقيمة n هو ٨٠٠٠ حرف).. والقيمة الافتراضية لطول النص في هذا النوع هي ٥٠ حرفا، وفي هذه الحالة لو حاولت كتابة نص طوله ٥١ حرفا فسيرفض سيكويل سيرفر ذلك، بينما لو كتبت نصا طوله ٨ حروف فسيتم حفظه فعلا في ٨ وحدات، ولن يتم إكماله بمسافات.. واضح أن هذا يوفر مساحة التخزين، ويعطيك مرونة في حالة تفاوت أطوال النصوص التي ستدخلها في العمود. | n<br>بحد أقصى | **varchar(n)** |

| | | |
|---|---|---|
| **varchar(MAX)** | | هذه الصيغة حالة خاصة من النوع السابق، وهي تجعل الحقل يستوعب نصا ضخما يصل إلى حجمه إلى ٢^٣١ -١ حرفا.. وفي هذه الحالة يكون حجم الحقل = عدد حروف النص المكتوب فيه + ٢.<br>ولا يمكن استخدام عمود من هذا النوع كمفتاح Key للجدول أو كمفتاح في أحد الفهارس Indexes. |
| **text** | ٢ مليار بحد أقصى | مشابه للنوع varchar في كل شيء، إلا أنه يسمح بحفظ نص مرن متغير الطول يمكن أن يصل إلى حوالي ٢ مليار حرف (٢^٣١ -١ حرف).. وأيضا لا يقبل هذا النوع حروف الكود الموسع Unicode. |
| **nchar(n)** | ٢×n | نص ثابت الطول طوله n يدعم الحروف الموسعة Unicode، لهذا يتم حفظ كل حرف في ٢ وحدة Byte 2، ولهذا فإن أقصى عدد مسموح به من الحروف في هذا النوع هو ٤٠٠٠ حرف.. هذا النوع مناسب للغة العربية. |
| **nvarchar(n)** | ٢+٢×n بحد أقصى | نص مرن متغير الطول يدعم الحروف الموسعة Unicode، بحد أقصى ٤٠٠٠ حرف.. هذا النوع مناسب للغة العربية. |
| **nvarchar(MAX)** | | هذه الصيغة حالة خاصة من النوع السابق، وهي تجعل الحقل يستوعب نصا ضخما يصل إلى حجمه إلى ٢^٣١ -١ وحدة ثنائية.. وفي هذه الحالة يكون حجم الحقل = ٢× عدد حروف النص المكتوب فيه + ٢. |
| **ntext** | ١ مليار بحد أقصى | نص مرن متغير الطول يدعم الحروف الموسعة Unicode، بحد أقصى حوالي مليار حرف (٢^٣٠ -١ حرف).. هذا النوع مناسب للغة العربية. |

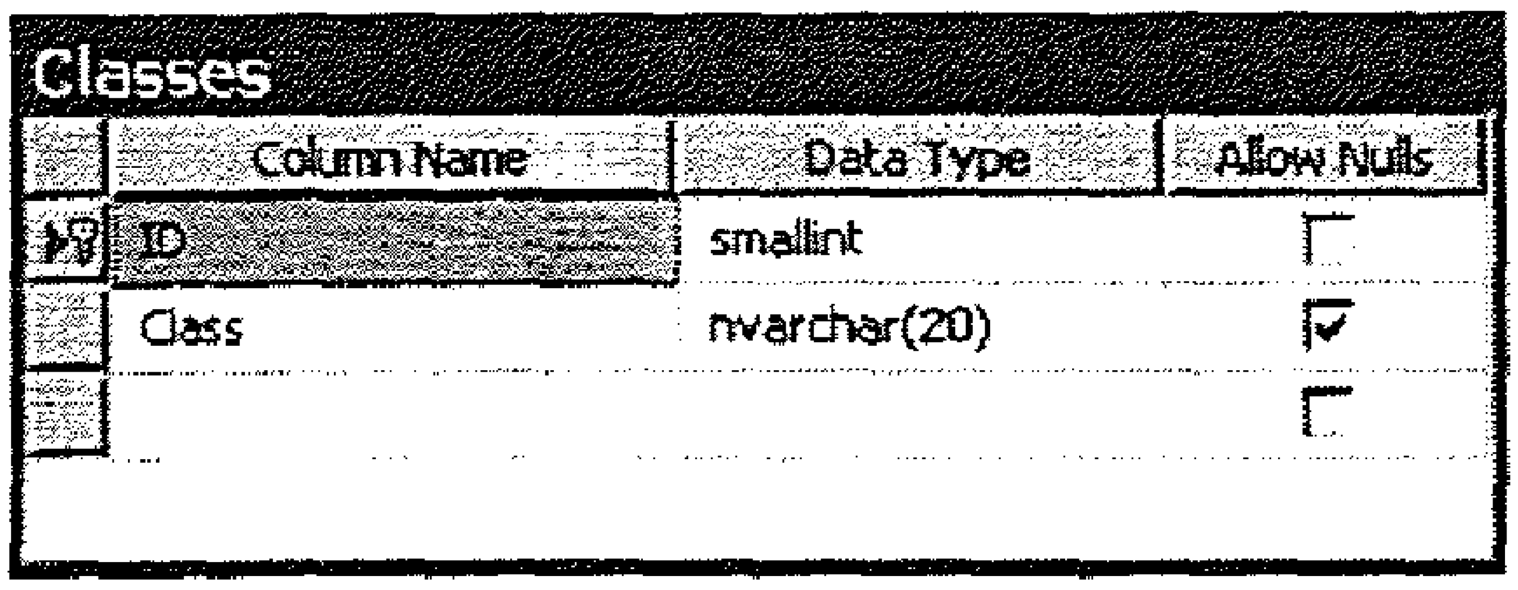
| | | |
|---|---|---|
| **binary(n)** | n +٤ | بيانات ثنائية ثابتة الطول، وبشرط ألا تزيد قيمة n عن ٨٠٠٠ وحدة ثنائية Bytes.. لو اخترت n = ٥٠ مثلا، وحفظت في إحدى الخانات بيانات تقل عن ٥٠ وحدة، فسيتم إكمال الباقي بالعدم Null (الحرف رقم ٠ في ترميز ASCII). |
| **varbinary(n)** | n +٤ كحد أقصى | بيانات ثنائية مرنة متغيرة الطول، وبحد أقصى ٨٠٠٠ وحدة ثنائية.. ولو حفظت بيانات طولها أقل من n، لا يتم إكمال الطول الناقص بأي شيء. |
| **Varbinary (MAX)** | | هذه الصيغة حالة خاصة من النوع السابق، وهي تجعل الحقل يستوعب بيانات يصل حجمها إلى ٢^٣١ -١ وحدة ثنائية.. وفي هذه الحالة يكون حجم الحقل = عدد الوحدات المكتوبة فيه + ٢. |
| **image** | ٢ مليار بحد أقصى | بيانات ثنائية مرنة متغيرة الطول، يمكن أن تصل إلى حوالي ٢ مليار وحدة ثنائية (٢^٣١ -١ Bytes). |
| **sql_variant** | ٨٠١٦ بحد أقصى | متغير عام يقبل باقي أنواع البيانات (الأرقام والنصوص)، ولا يستثنى من هذا إلا الأنواع التالية: text, ntext, timestamp, varchar(MAX) |
| **uniqueidentifier** | ١٦ | معرف عام متفرد GUID وهو رقم سداسي عشري Hexadecimal غير قابل للتكرار. |
| **xml** | ٢ مليار بحد أقصى | يتيح لك هذا النوع تخزين صفحات XML أو أجزاء منها في خانات العمود. |
| **Timestamp** | ٨ | مصفوفة من ٨ خانات ثنائية، تقوم بإنتاج أعداد ثنائية Bytes، بحيث تكون المصفوفة متفردة عبر كل الجدول، وذلك لاستخدامها في تمييز النسخ المحدثة من كل صف Version-Stamping. |
| **Udt** | | نوع يعرفه المستخدم User-Defined Type. |
| **Structured** | | نوع خاص للبيانات المركبة في صورة جدول Table-valued، وسنتعرف عليه لاحقا. |

| ملحوظة ١: |
| --- |
| عند استخدام أكثر من عمود متغير القيمة (ينتهي بقوسين بينهما طولـــه (n))، يجب ألا يزيد الطول الإجمالي للسجل ككل عن ٨٠٠٠ وحـــدة ثنائيـــة Byte، وإلا حدث خطأ.. ولتجاوز هذا الحجم المحـــدود للـــسجل، عليـــك اســـتخدام المتغيرات Text وnText و image أو المتغيرات الجديدة التي ظهرت فـــي سيكويل سيرفر ٢٠٠٥، وهي تحمل نفس أسماء المتغيرات محددة الطول، لكنها تنتهي بالقيمة القصوى (Max)، وبذلك تتيح لك التعامل مع بيانـــات يتجـــاوز طولها ٢ مليار وحدة ثنائية Byte.. كما تتيح لك هذه المتغيـــرات إمكانيـــات جديدة في كتابة وقراءة البيانات بطريقة تتابعية Sequential، كمـــا ســـنرى لاحقا. |

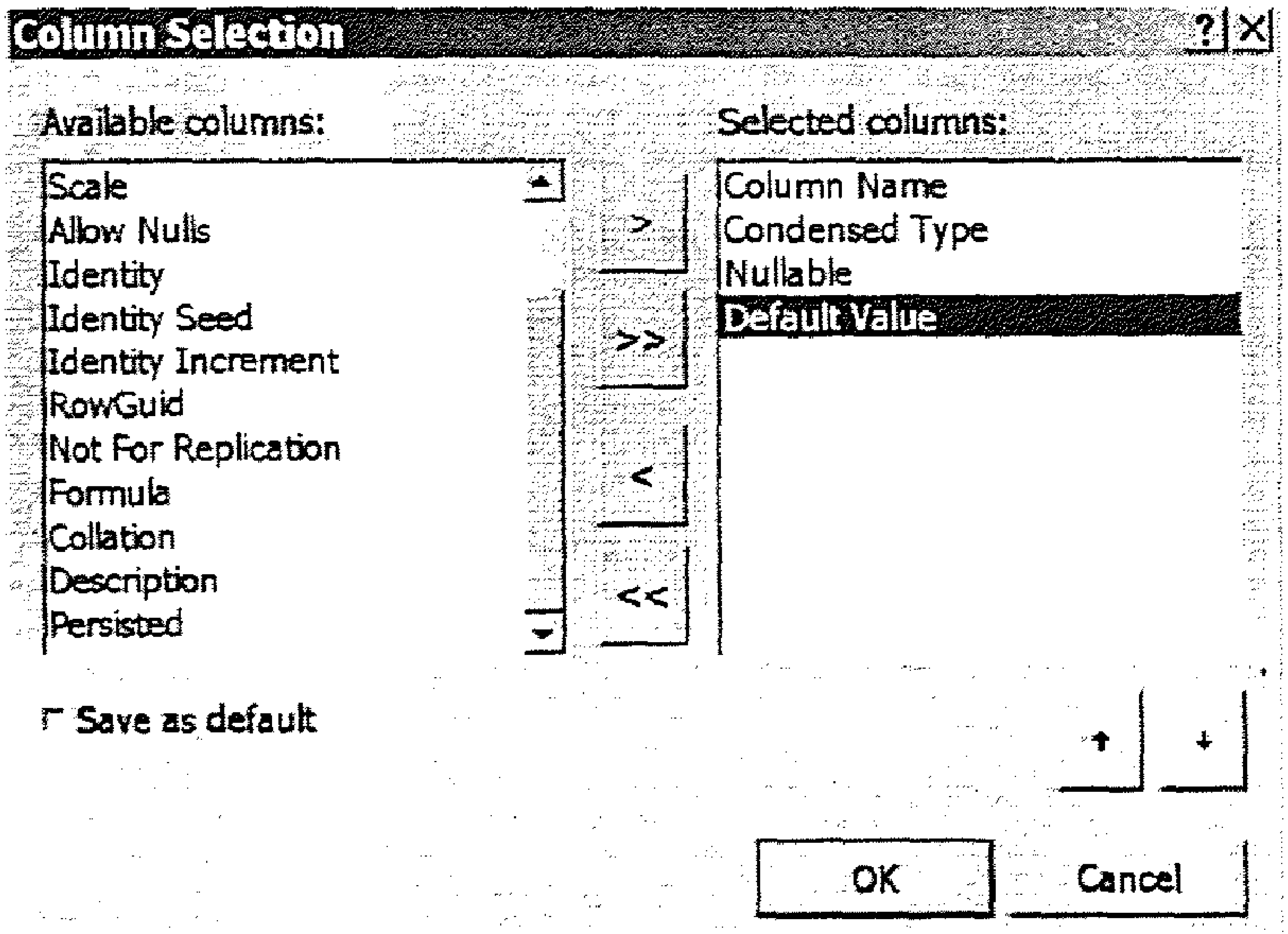
| ملحوظة٢: |
| --- |
| لجعل أحد الحقول ترقيما تلقائيا، اختر له نوعا رقميا مناسبا، وفي الجزء السفلي من النافذة اضغط العلامة + المجـــاورة للعنـــصر Identity Specification لإسدال خصائصه، واجعل للخاصية (Is Identity) القيمة Yes لجعل الحقـــل معرفا متفردا.. هذا سيغير قيمة كل من الخاصيتين "معدل الزيـــادة" Identity Increment وبذرة الزيادة Identity Seed إلى القيمة ١.. ولو أردت أن تبدأ الترقيم التلقائي من العدد ١٠٠ مثلا، فضع في الخاصية Identity Seed القيمة ١٠٠.. ولو أردت أن يزيد ترقيم كل حقل عن السابق له بمقدار ٣، فضع فـــي الخاصية Identity Increment القيمة ٣.<br>ولجعل الحقل مفتاحا أساسيا، اضغط الهامش الأيسر المجاور له بـــزر الفـــأرة الأيمـــن، ومـــن القائمـــة الموضـــعية اضـــغط الأمـــر Set Primary Key.. ولو أردت أن تتراجع عن جعل الحقل مفتاحا أساســـيا، فاضغطه بزر الفأرة الأيمن مجددا، ومن القائمة الموضـــعية اضـــغط الأمـــر Remove Primary Key. |

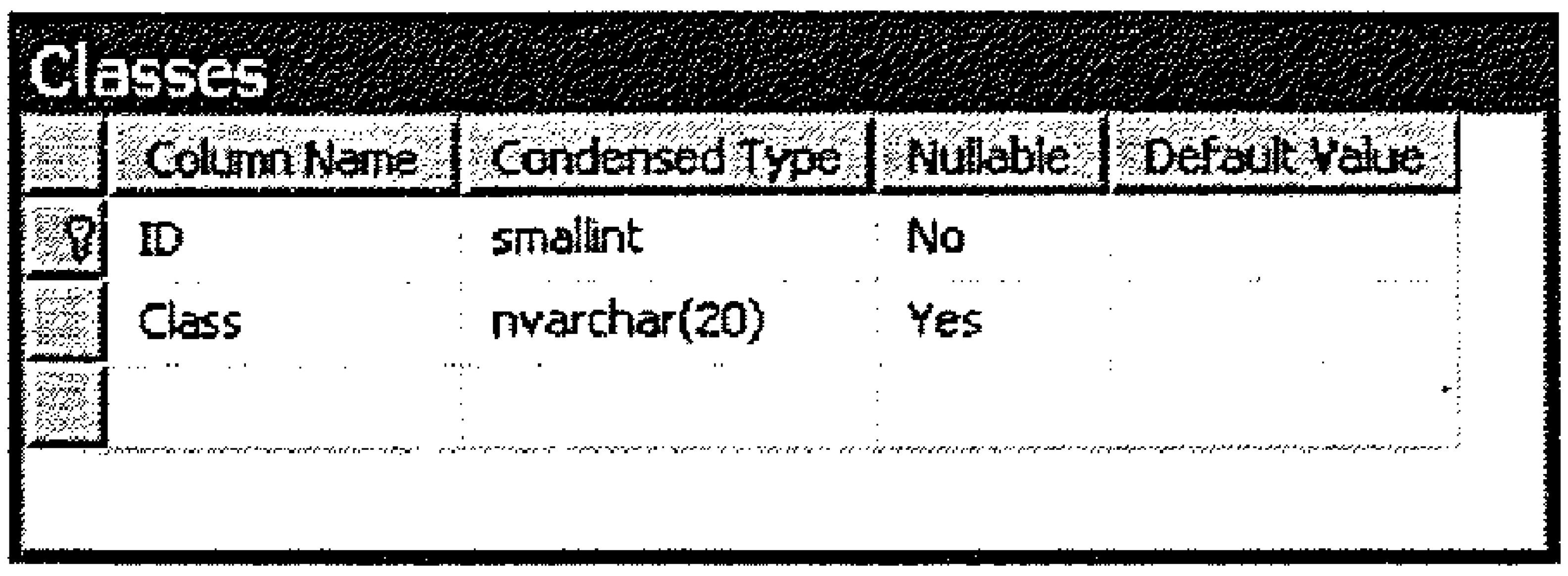
## إنشاء العلاقات في سيكويل سيرفر:

في متصفح الكائنات Object Explorer أسدل عناصر قاعدة بيانات الكتب Books، وبزر الفأرة الأيمن اضغط "مخططات قاعدة البيانات" Database Diagrams، ومن القائمة الموضعية اضغط الأمر New Database Diagram.. سيؤدي هذا إلى عرض نافذة اختيار الجداول التي ستستخدمها في رسم المخطط.. حدد جدولا أو أكثر بالفأرة، ثم اضغط الزر Add لإضافتها إلى صفحة المخطط، ثم اضغط Close لإغلاق النافذة والانتقال إلى صفحة المخطط.. هذه النافذة تشبه نافذة إنشاء العلاقات في آكسيس، ولكنها تمنحك ميزات إضافية، فبإمكانك تغيير أسماء أعمدة أي جدول بمجرد ضغطها وتحريرها في هذا المخطط، كما يمكنك ضغط أي حقل بزر الفأرة الأيمن وضغط الأمر Set Primary Key من القائمة الموضعية لجعل هذا الحقل مفتاحا أساسيا.



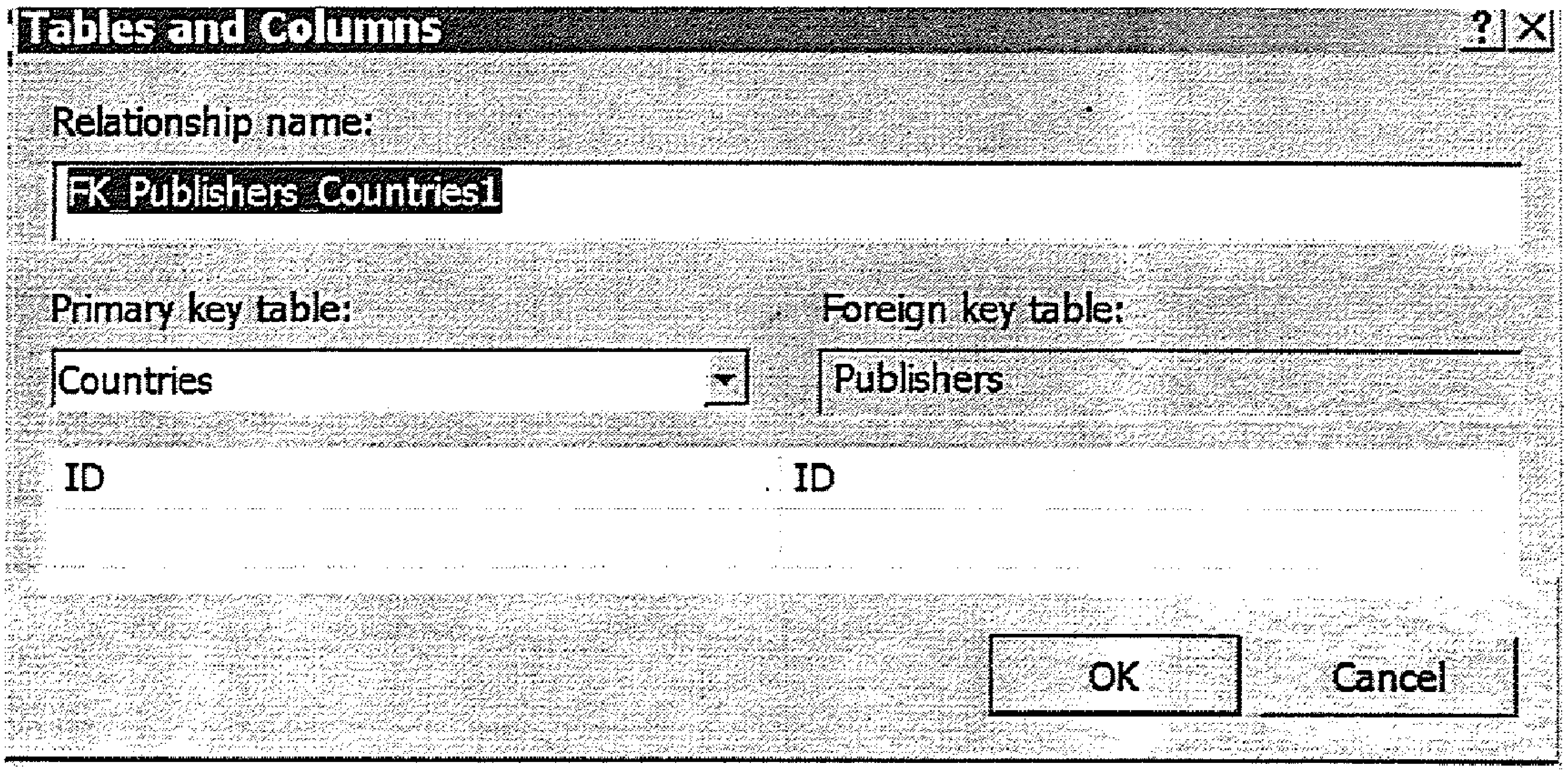

وفي الوضع الطبيعي، يعرض المستطيل الممثل لكل جدول أسماء أعمدته، لكنه تستطيع أن تعرض أيضا تفاصيل كل عمود (اسمه، ونوع بياناته، والسماح بتركه فارغا)، وذلك بضغط مستطيل الجدول بزر الفأرة الأيمن لعرض القائمة الموضعية، ثم ضغط القائمة الفرعية Table View، وضغط الأمر Slandered.. كما يمكنك اختيار طرق عرض أخرى، كأن تختار عرض المفاتيح الأساسية الموجودة في الجدول فقط Keys، أو اختيار عرض اسم الجدول فقط Name Only، بل يمكنك تصميم طريقة عرض خاصة بك Custom.. مبدئيا ستعرض هذه الطريقة تفاصيل الجدول الأساسية، ولكن يمكنك إضافة أية تفاصيل أخرى بضغط الزر Modify Custom، واستخدام النافذة التي ستظهر لنقل التفاصيل التي تريد عرضها من القائمة اليسرى إلى القائمة اليمنى، وذلك بضغط الزر > .



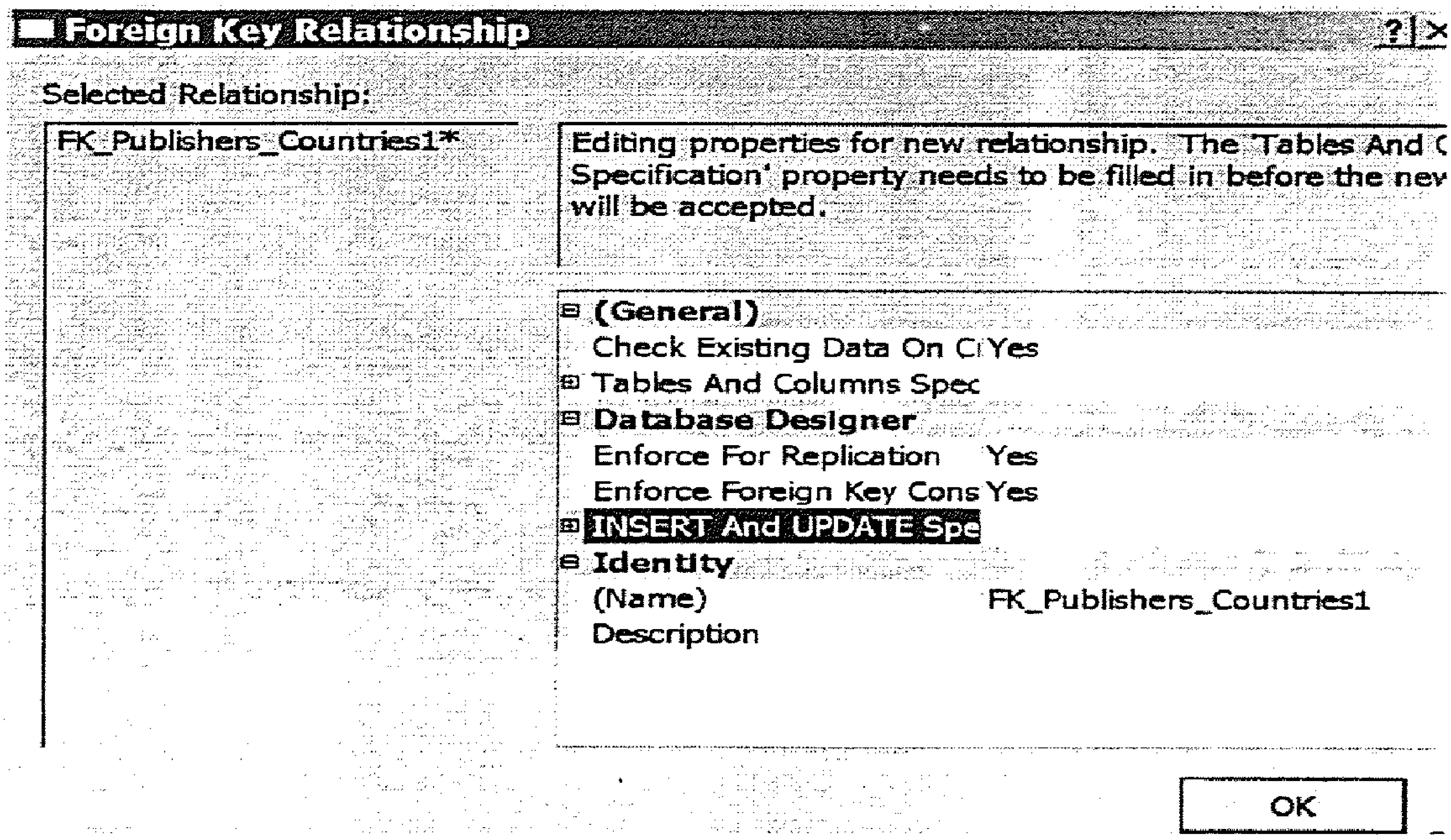



وكما ترى في الصورة أعلاه، فقد أضفنا عمود القيمة الافتراضية للعمود، وهذا سيؤدي إلى عرض مخطط الجدول كما في الصورة.. ولو أردت العودة إلى طريقة العرض التقليدية، فاضغط الأمر Column Names لعرض أسماء الأعمدة فقط.

وكما في آكسيس، يمكنك سحب أي حقل من جدول وإلقاؤه على أي حقل في جدول آخر لتكوين علاقة بين الجدولين.. سيؤدي هذا إلى ظهور نافذة توضح الجدولين والعمودين الداخلين في العلاقة، لتتيح لك تغييرهما إن أردت:




وبعد أن تضغط OK ستظهر نافذة توضح اسم وخصائص العلاقة بين الجدولين:



في هذه النافذة يمكنك إدخال اسم العلاقة، وتحديد مواصفاتها، بطريقة مماثلة للتي اتبعناها في Access.. لاحظ أن اسم العلاقة الافتراضي يكون على الصيغة: FK_T1_T2n، حيث FK اختصار "المفتاح الفرعي" Foreign Key، و T1 هو اسم الجدول الأول (الذي يحتوي على المفتاح الأساسي)، و T2 هو اسم الجدول الثاني (الذي يحتوي على المفتاح الفرعي)، و n هو رقم العلاقة، وهو يبدأ بالرقم ١، ويمكنك حذفه لو كانت هناك علاقة واحدة فقط بين الجدولين.. وعلى كل حال، يمكنك تغيير هذا الاسم إلى أي اسم آخر يناسبك، بتغيير الخاصية (Name) الموجودة أسفل النافذة.. كما تحتوي هذه النافذة على الخصائص التالية:

- **Check existing data on creation or re-enabling:**

إذا جعلت قيمة هذه الخاصية Yes (وهي القيمة الافتراضية)، فسيتم التحقّق من صحّة البيانات الموجودة حاليا في الجدولين، للتأكد من أنها تتفق مع شروط العلاقة التي سيتم إنشاؤها.

- **Enforce for replication:**

إذا جعلت قيمة هذه الخاصية Yes (وهي القيمة الافتراضية)، فسيتم إنشاء العلاقة في قاعدة البيانات الجديدة عند نسخ قاعدة البيانات الحالية.

- **Enforce foreign key constraint:**

إذا جعلت قيمة هذه الخاصية Yes (وهي القيمة الافتراضية)، فسيتم التحقق من قيد المفتاح الفرعي، وهو يتأكد من عدم وجود بيانات في الجدول الفرعي مرتبطة بمفتاح غير موجود في الجدول الأساسي.. بعبارة أخرى: هذا القيد يدفع قاعدة البيانات لتحقيق التكامل المرجعي Referential Integrity بين الجدولين، لهذا لو حاولت حذف سجل من الجدول الأساسي وكان مرتبطا ببيانات في الجدول الفرعي، فسيحدث خطأ ولن يتم تنفيذ هذه العملية.. ولو أردت أن تنفذ عملية الحذف، فعليك أولا أن تحذف السجلات ذات الصلة من الجدول الفرعي، ثم تحذف بعدها السجل الأساسي المرتبط بها من الجدول الأساسي.

- **INSERTs and UPDATEs Specifications:**

اضغط العلامة + المجاورة لهذه الخاصية، لتظهر لك الخاصيتان التاليتان:

  - **قاعدة الحذف Delete Rule:**

  تتحكم هذه الخاصية في ماذا يحدث عندما يتم حذف بعض السجلات الداخلة في العلاقة.

  - **قاعدة التحديث Update Rule:**

  تتحكم هذه الخاصية في ماذا يحدث عندما يتم تحديث بيانات بعض السجلات الداخلة في العلاقة.

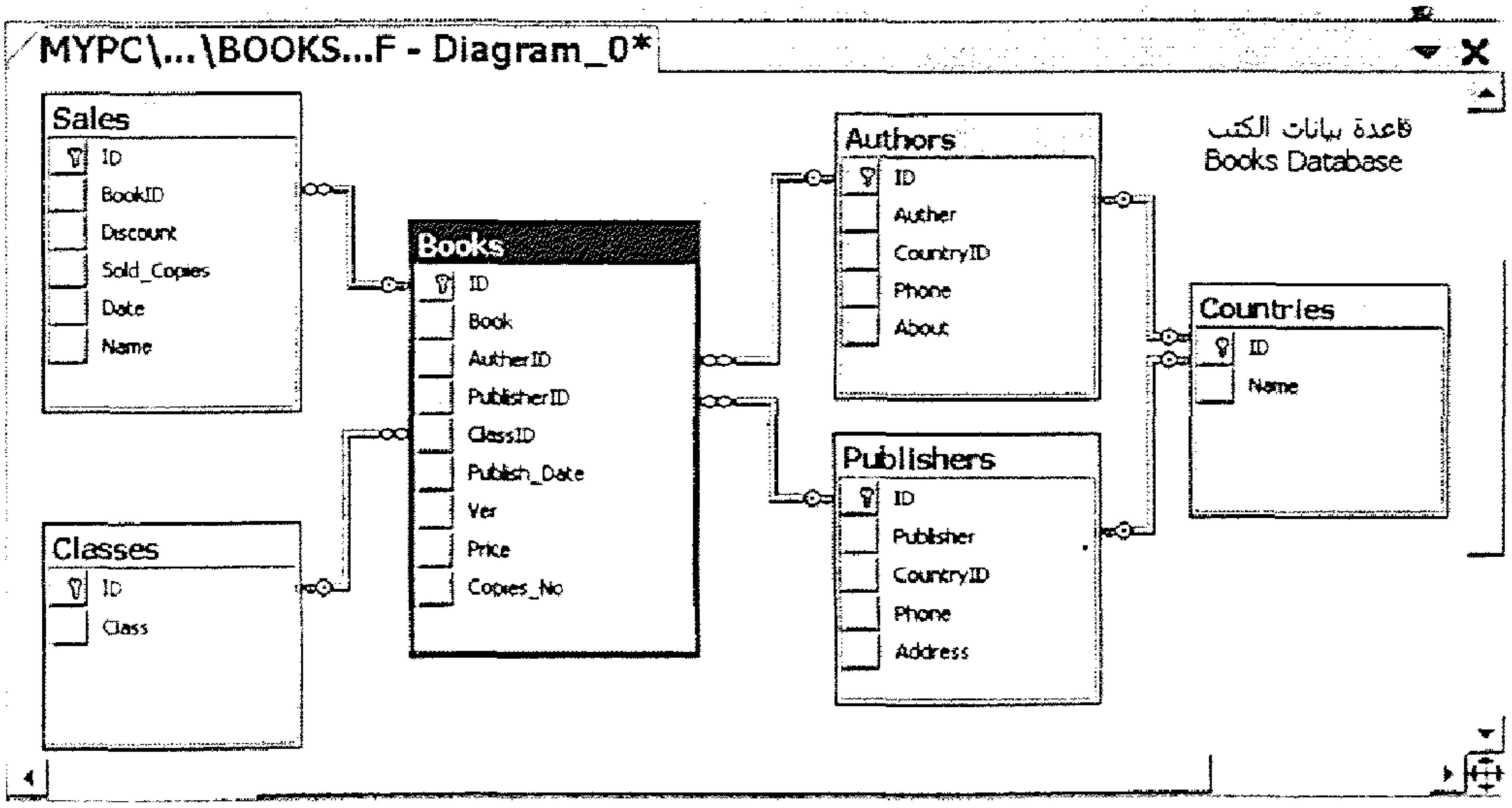
والجدول التالي يوضح القيم المختلفة لهاتين الخاصيتين:

| | |
|---|---|
| No Action | عدم اتخاذ أي إجراء.. هذه هي القيمة الافتراضية في النافذة. |
| Cascade | عند تحديث المفتاح الأساسي، يتم تلقائيا تحديث خانة المفتاح الفرعي في السجلات المرتبطة به في الجدول الفرعي.<br>وعند حذف سجل المفتاح الأساسي، يتم تلقائيا حذف جميع السجلات المرتبطة به في الجدول الفرعي. |
| Set Null | عند حذف أو تحديث المفتاح الأساسي، يتم وضع قيمة منعدمة DBNull في خانة المفتاح الفرعي في السجلات المرتبطة به في الجدول الفرعي. |
| Set Default | عند حذف أو تحديث المفتاح الأساسي، يتم وضع القيمة الافتراضية في خانة المفتاح الفرعي في السجلات المرتبطة به في الجدول الفرعي. |

اضغط OK لإغلاق النافذة، حيث سيظهر خط بين الجدولين، يمثل هذه العلاقة.

| **ملاحظة:** |
|---|
| لكي تنشئ علاقة بين حقلين، يجب أن يكون لهما نفس نوع البيانات ونفس الطول، وإلا فسيرفض سيكويل سيرفر إنشاء العلاقة.. لهذا تأكد من أن للمفاتيح الأساسية والمفاتيح الفرعية نفس النوع لتصلح للدخول في العلاقات. |

وتستطيع تغيير شكل خط العلاقة بمرونة، بمجرد سحب أي جزء منه بالفأرة، كما أن طرف بدايته يأخذ شكل مفتاح للإشارة إلى حقل المفتاح الأساسي، بينما يأخذ طرفه الآخر شكل علامة "ما لا نهاية" للإشارة إلى علاقة "واحد بمتعدد".



ولمراجعة أو تعديل خصائص العلاقة، اضغط خطّ العلاقة بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اضغط الأمر Properties.. ستظهر نافذة الخصائص وبها خصائص العلاقة، بحيث يمكنك تعديلها كيفما شئت.. أما لو شئت حذف العلاقة، فاضغط الخطط الممثل لها بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Delete Relationship From database، وأكد رسالة الحذف الذي ستظهر لك بضغط Yes.

وتتيح لك صفحة المخططات بعض التسهيلات الإضافية:

- إذ يمكنك مثلا أن تغير نسبة العرض Zoom باستخدام القائمة المنسدلة الموجودة على شريط الأدوات، أو باستخدام القائمة الفرعية Zoom التي تعرضها القائمة الموضعية عند الضغط بزر الفأرة الأيمن في أي منطقة خالية من صفحة المخططات.
- كما يمكنك أن تختار أيضا من القائمة الموضعية إضافة تعليق نصي، وذلك بضغط الأمر Add Text Annotation، حيث سيؤدي هذا إلى إدراج مربع نص على صفحة المخطط لتكتب فيه ما تشاء، مع قدرتك على سحب إطاره بالفأرة لتغيير حجمه أو موضعه، ولو ضغطت داخله بزر الفأرة الأيمن، فيمكنك اختيار تغيير الخط من القائمة الموضعية وذلك بضغط الأمر Set Text Annotation Font.
- ويمكنك عرض لافتة على خط كل علاقة توضح اسم العلاقة، وذلك بضغط الأمر Show Relationship Labels.
- كما يمكنك عرض خط أزرق يمثل حدود صفحة الطباعة لكي تضمن عدم تجاوز الجداول له في حالة رغبتك في طباعة المخطط، وذلك بضغط الأمر View Page Breaks.
- وبعد أن تنتهي من تصميم المخطط، استخدم الأمر Copy Diagram to Clipboard لنسخه كصورة إلى لوحة قصاصات الويندوز، ومن ثم يمكنك لصقه في أي برنامج تحرير صور مثل Paint أو حتى Word لتستطيع طباعته أو استخدامه في وثائقك.
- كما يمكنك طباعة المخطط مباشرة باستخدام الأمرين Page Setup و Print من القائمة الرئيسية File.

والآن قم بحفظ هذا المخطط، وذلك بضغط Ctrl+S حيث ستظهر لك نافذة تسألك عن اسم المخطط.. سمّه Diagram1 مثلا.. ستظهر لك رسالة تنذرك بأنّ هناك تغييرات ستحدث لبعض الجداول لتمنحك الفرصة للتراجع.. لو أردت ألا تظهر لك هذه الرسالة في كل مرّة، فأزل علامة الاختيار من مربّع الاختيار

Warn About Tables Affected.. اضغط Yes لإتمـام العمليّـة وإغـلاق الرسالة.

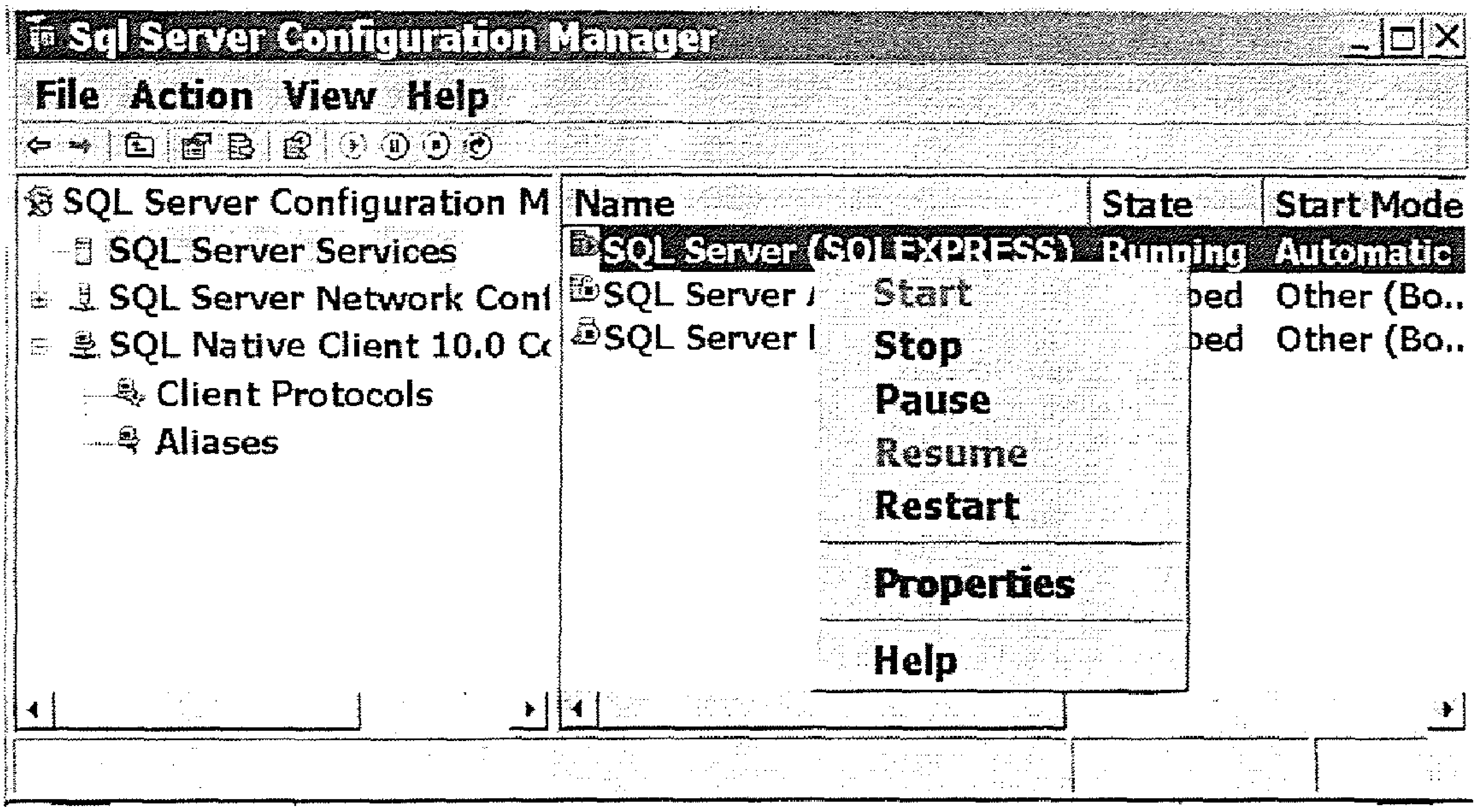
لاحظ أنك تستطيع إنشاء أكثر من مخطط وحفظها بأسماء مختلفة.. هذا يتيح لـك تقسيم العلاقات بين الجداول على أكثر من مخطط، وذلك إذا كانت جداول قاعـدة البيانات كثيرة جدا ومن العسير وضعها كلها معا في مخطط واحد.

## التحكم في خادم سيكويل:

يمكنك إيقاف خادم سيكويل عن العمل وإعادة تشغيله في أي وقت، وذلك باستخدام مدير تهيئة خادم سيكويل SQL Server Configuration Manager.. لفـتح هذا البرنامج، اضغط:

Start\Programs\Microsoft SQL Server 2008\
Configuration Tools\ SQL Server Configuration Manager

وفي النافذة التي ستظهر لك، اضغط العنـصر SQL Server Services مـن الشجرة الموجودة على اليسار:




سيعرض هذا ثلاث خدمات من خدمات سيكويل سيرفر، ما يهمنا منها هي خدمة سيكويل سيرفر نفسها SQL Server (SQLExpress).. لـو ضـغطت هـذه الخدمة بزر الفأرة الأيمن، فستعرض لك القائمة الموضعية الأوامر التي تتيح لك إيقاف Stop أو تشغيل Start أو إعادة تشغيل Restart خادم سيكويل.. وعليـك أن تنتبه إلى أن إيقاف هذا الخادم سيمنعك من التعامل مـع أي قاعـدة بيانـات موجودة عليه، لهذا عليك أن تتأكد أنه يعمل قبل أن تحاول الاتصال به مـن أي برنامج خارجي.. لاحظ أيضا أن حالة الخادم تظل ثابتة (متوقفا كـان أم يعمـل)

عند إغلاق الحاسب وإعادة تشغيله.. لهذا إن كنت أوقفته على سبيل التجريب، فلا تنسَ إعادة تشغيله مجددا، واتركه يعمل في سلام ☺.

## توصيل وفصل قاعدة بيانات:

لفصل قاعدة بيانات الكتب من العمل على الخادم، اضغط اسمها بزر الفأرة الأيمن في متصفح الكائنات Object Explorer، ومن القائمة الموضعية اضغط القائمة الفرعية Tasks، ومنها اضغط الأمر Detach، وعند ظهور نافذة تأكيد الأمر اضغط OK.. قد يحدث خطأ يخبرك بأن هناك عمليات اتصال بقاعدة البيانات حاليا.. في هذه الحالة اضغط علامة الاختيار في مربعي الاختيار المجاورين لاسم القاعدة في النافذة، لإنهاء عمليات الاتصال بها، واضغط زر OK مجددا.

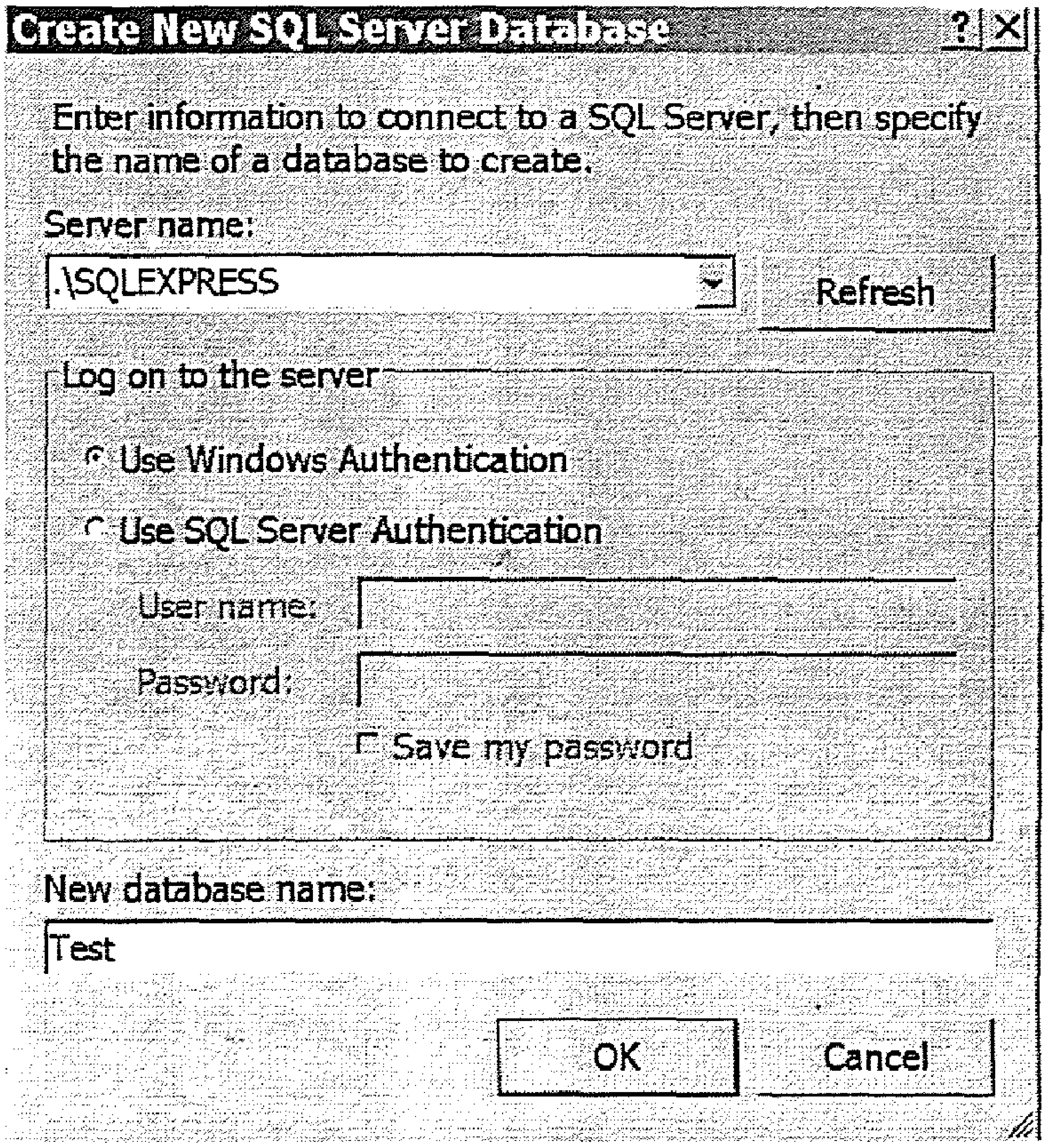
ولإعادة توصيل هذه القاعدة مرة أخرى، اضغط العنصر Databases في متصفح الكائنات بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Attach، وفي نافذة توصيل قاعدة البيانات اضغط الزر Add واختر ملف قاعدة البيانات من الموضع الذي حفظتها فيه على جهازك.. ويمكنك أن تضيف أكثر من قاعدة بيانات إلى القائمة لتوصيلها كلها مرة واحدة.. بعد هذا اضغط OK لتوصيل قواعد البيانات التي اخترتها، حيث ستظهر أسماؤها في متصفح الكائنات.

## التعامل مع قواعد بيانات سيكويل سيرفر من دوت نت:

تقدم دوت نت ٢٠١٠ دعما كاملا للتعامل مع قواعد بيانات سيكويل سيرفر ٢٠٠٨.. ولا يتوقف الأمر على الاتصال بقاعدة البيانات وقراءة البيانات منها، بل إن الأمر يتعدى ذلك إلى إنشاء قواعد البيانات وتصميم جداولها وعلاقاتها وكل كائناتها من داخل بيئة التطوير المتكاملة IDE الخاصة بدوت نت، بل إنك ستشعر أنه يكاد لا يوجد فارق بين طريقة التعامل مع قاعدة البيانات في دوت نت، وبين التعامل معها في مدير سيكويل سيرفر SQL Server Management Studio.. الفارق الوحيد هو أنك هنا ستتعامل مع متصفح الخوادم Server Explorer بدلا من متصفح الكائنات Object Explorer!

## إنشاء قاعدة بيانات سيكويل سيرفر في دوت نت:

افتح متصفح الخوادم Server Explorer من القائمة View، واضغط بزر الفأرة الأيمن العنصر Data Connections، ومن القائمة الموضعيّة اضغط الأمر Create New SQL Server Database.. سيظهر لك مربّع الحوار الموضّح في الصورة التالية:




في هذه النافذة مطلوب منك أن تكتب اسم خادم سيكويل الذي ستنشئ عليه قاعدة البيانات.. اكتب اسم الخادم MYPC\SQLEXPRESS.

| **ملحوظة:** |
| --- |
| هناك اختصار متعارف عليه يستخدم بدلا من اسم الخادم المحلي، وهو الكلمة (local) موضوعة بين قوسين، أو النقطة "." .. الحكمة في هذا أنك تستطيع تغيير اسم جهازك من لوحة تحكم الويندوز Control Panel، وفي هذه الحالة ستحدث أخطاء في برامجك التي تستخدم الاسم القديم، لهذا فمن الأذكى أن تستخدم الاسم .\SQLEXPRESS بدلا من MYPC\SQLEXPRESS . |

ولو كان الخادم محميا باسم مستخدم وكلمة سر، فاضغط الاختيار Use SQL Server Authentication، حيث سيتم تفعيل مربعي النص الخاصين بكلمة

المرور واسم المستخدم.. ولكن نظرا لأننا نستخدم الخادم المحلي بدون حمايـــة، فاترك الاختيار الأول Use Windows NT Integrated Security.

أخيرا، اكتب اسم قاعدة البيانات التي تريد إنشائها، وذلك في مربع النص الموجود أسفل النافذة.. اكتب مثلا الاسم Test واضغط OK لإغلاق النافذة.

ستجد أنّ الاسم mypc\sqlexpress.Test.dbo قد ظهر في متصفح الخوادم.

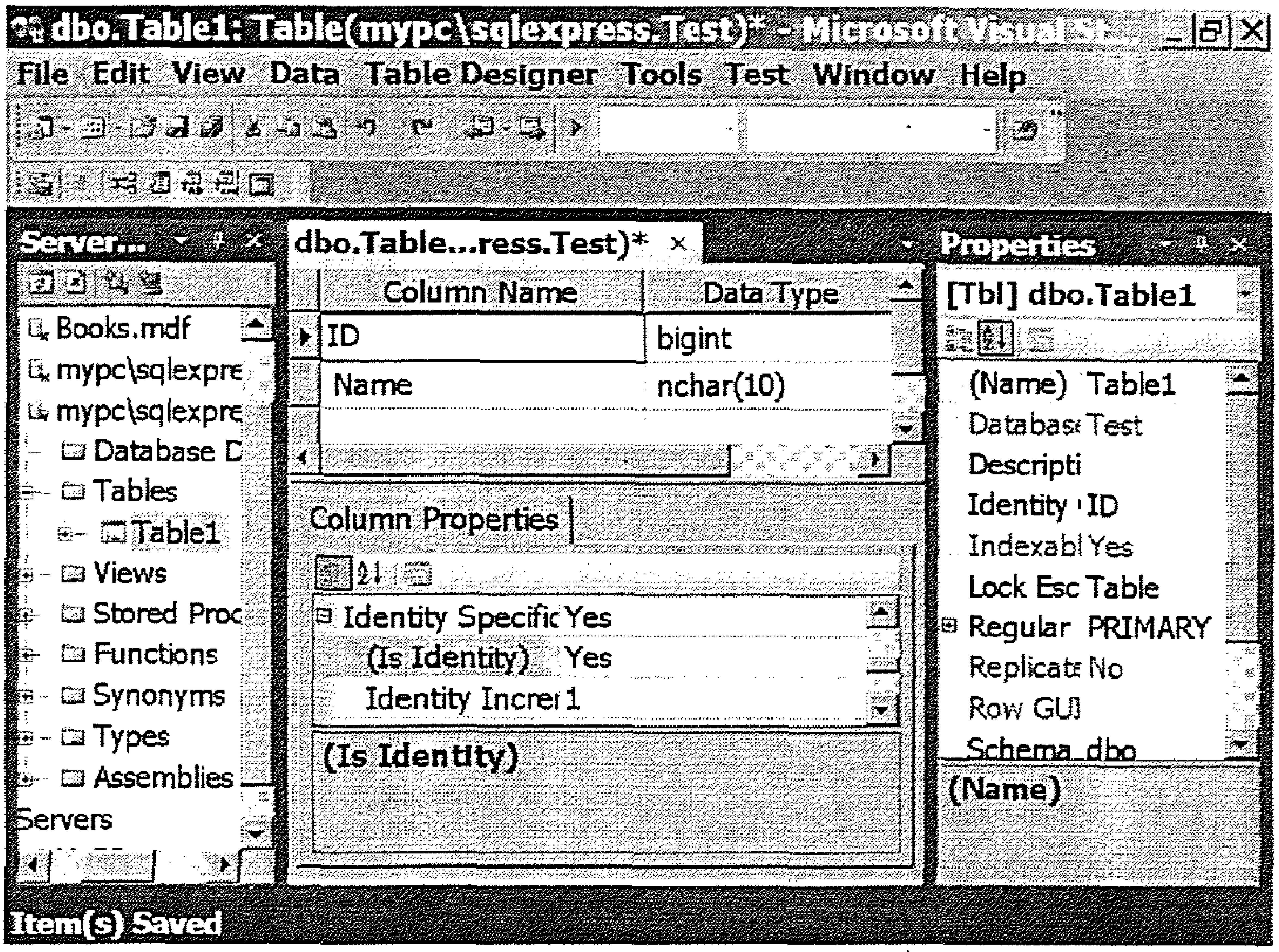
وسيتمّ إنشاء قاعدة البيانات Test.mdf على المسار:

C:\Program Files\Microsoft SQL Server\
MSSQL10.SQLEXPRESS\MSSQL\DATA

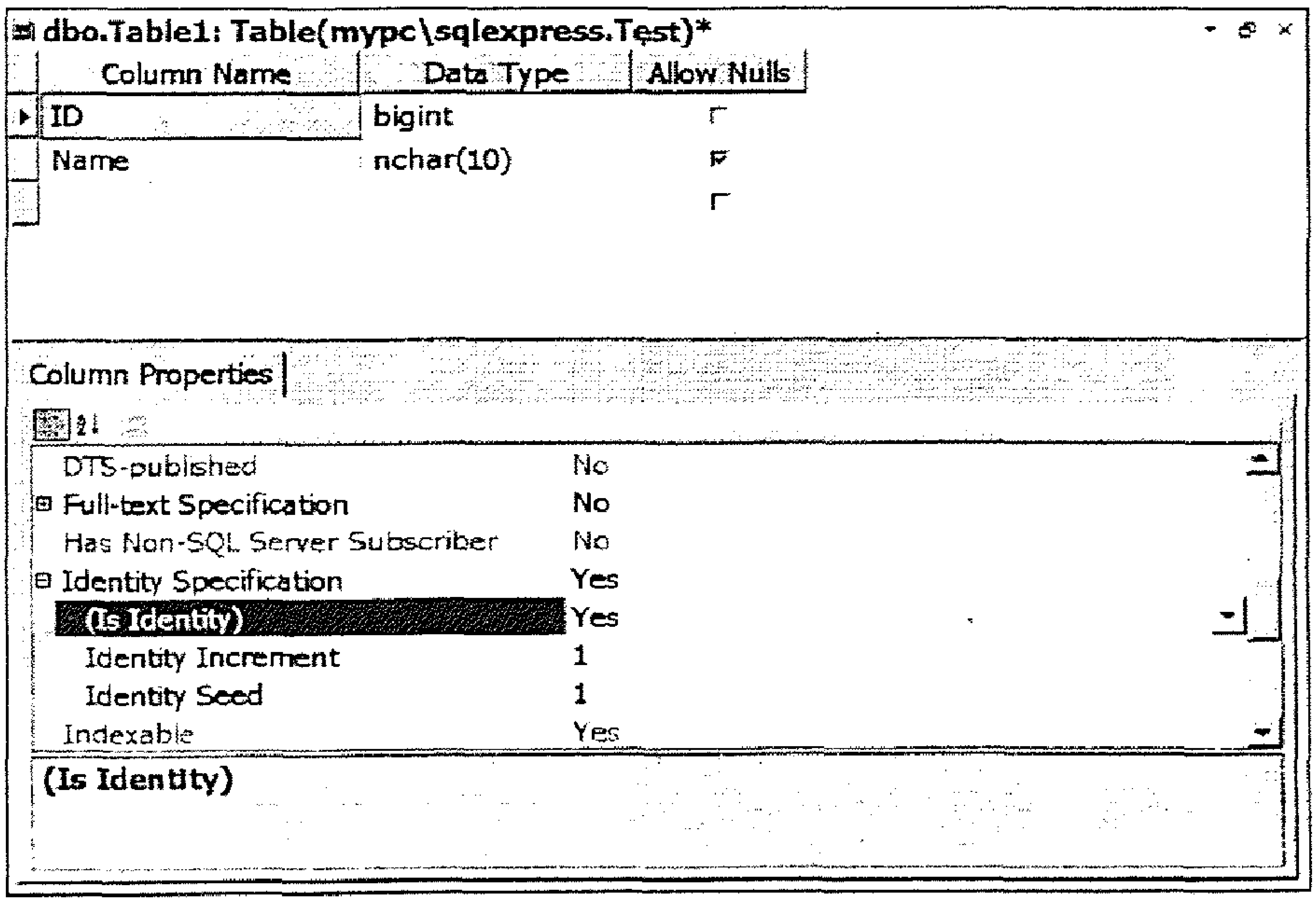
ولو شئت أخذ نـسخة احتاطيّـــة منهـــا، فـــلا تـــنسَ أن تتـــسخ أيـــضا الملـــف Test_log.LDF، ولو رفض الويندوز تنفيذ النسخ، فأغلق البرامج المتصلة بقاعدة البيانات، وإن اضطررت فافصل قاعدة البيانات من العمل على الخادم.

| ملحوظة: |
| --- |
| لو ضغطت القاعدة Test في متصفح الخوادم Server Explorer بزر الفأرة الأيمن، وضغطت الأمر Delete من القائمة الموضعيّة، فسيتم حذف الاتصال بقاعدة البيانات من متصفح الخوادم، لكن لن يتم حذف قاعدة البيانات نفـسها، ومن ثم يمكنك إعادة الاتصال بها مرة أخـــرى مـــن جديـــد بالطريقـــة التـــي سنوضحها لاحقا.. لاحظ أن هذا يختلف تماما عما سيحدث في مدير ســيكويل SSMS، فضغط الأمر DELETE سيعرض نافذة حـــذف قاعـــدة البيانـــات نفسها، لذا عليك الحذر، استخدام الأمر Tasks\Detach لفصل قاعدة البيانات. |

والآن، أسدل عناصر قاعدة البيانات Test في متصفح الخوادم.. لن تشعر بأيـــة غربة، فأنت تشاهد نفس العناصر التي شاهدتها في مدير سيكويل من قبـــل، ولا تحتاج لأن يعلمك أحد كيف تنشئ الجداول أو العلاقات، فلا يوجـــد أي اخـــتلاف يذكر!.. لا أظنك تعتبره اختلافا جذريا، لو أخبرتك أن نقر اسم الجـــدول مـــرتين بالفأرة يفتح صفحة تصميمه، أو أن الأمر الذي يفعل هذا من القائمة الموضـــعية اسمه "فتح تعريف الجدول Open Table Definition" بدلا من الأمر Design في مدير سيكويل، أو أن عرض وتحرير بيانات الجدول يـــتم باســـتخدام الأمـــر Show Table Data بـــدلا مـــن الأمـــرين: Select Top 1000 rows و Edit Top 200 Rows!.. لاحظ أن نافذة عرض بيانات الجدول التي تظهـــر بعد ضغط الأمر Show Table Data هي جزء النتائج Result Pane في نافذة باني الاستعلام Query Builder، ويمكنك عرض أي جزء آخر بالضغط بـــزر الفأرة في أي موضع، ومن القائمة الموضعية تـضغط القائمـــة الفرعيـــة Pane وتختار الجزء الذي تريد عرضه.




لاحظ أنك تستطيع تغيير طريقة عرض النوافذ في دوت نت من العرض المتلاحم Dock إلى العرض الحر.. مثلا، لو نقرت الشريط العلوي لنافذة تصميم الجدول مرتين بالفأرة، فستتحول إلى نافذة مستقلة يمكنك تكبيرها وتصغيرها وتحريكها خارج بيئة دوت نت كما تريد:

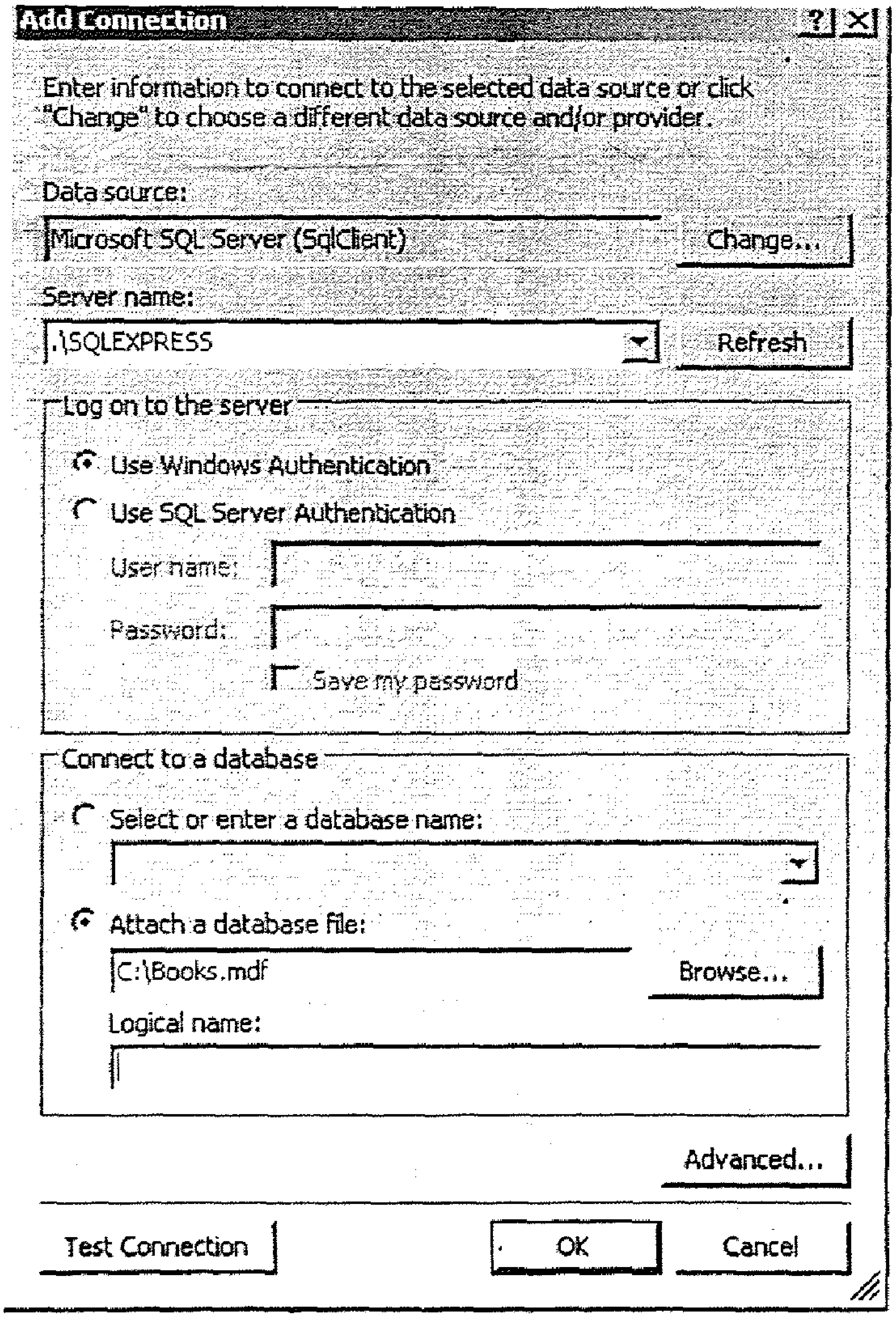



ولو أردت إعادتها إلى وضعها القديم، فاضغط شريط النافذة العلوي بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Dock.

## الاتصال بقواعد بيانات SQL Server:

نريد الآن الاتصال بقاعدة البيانات Books.mdf التي أنشأناها بمدير سيكويل.. في هذه الحالة سنتبع نفس الخطواتِ التي اتبعناها عندما أردنا الاتصال بقاعدة البيانات Books.mdb التي أنشأناها في Access:

- اضغط بزر الفأرة الأيمن العنصر Data Connections في متصفح الخوادم، ومن القائمة الموضعيّة اضغط الأمر "إضافة اتصال Add Connection".
- في مربّع حوار "اختيار مصدر البيانات Choose data source"، اختر العنصر: Microsoft SQL Server Database File واضغط OK.
- ستظهر لك النافذة الموضحة في الصورة:



في النصف العلوي من النافذة، اكتب اسم خادم سيكويل واختر طريقة تحقيق الهوية.
وفي النصف السفلي من النافذة، اختر قاعدة البيانات التي تريد التعامل معها.. إذا كانت هذه القاعدة تعمل على الخادم فعلا في هذه اللحظة، فاختر الخيار Select or enter a database file واكتب اسم قاعدة البيانات، أو اختره من القائمة المنسدلة.. أما إذا لم تكن قاعدة البيانات متصلة بالخادم في هذه اللحظة، فاختر Attach a database file، واضغط الزر Browse، واختر ملف قاعدة البيانات (وليكن Books.mdf) من المجلد الذي حفظته فيه على جهازك.. ويمكنك أن تكتب في الخانة Logical Name أي اسم تريده لتسمية الاتصال به، وليس شرطا أن يكون له نفس اسم قاعدة البيانات.. أو يمكنك ترك هذه الخانة فارغة.

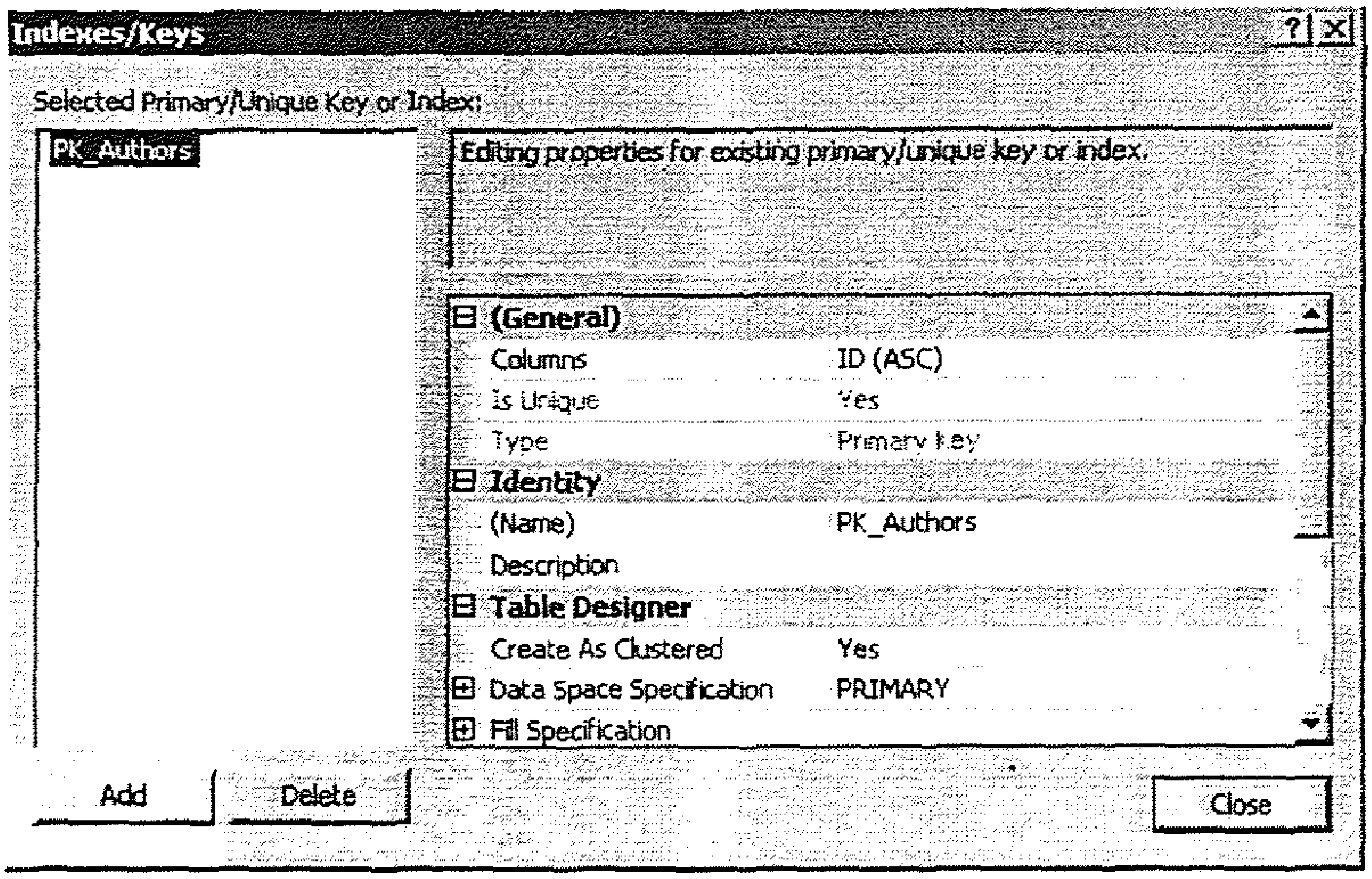
- اضغط الزر Test لاختبار الاتصال.. إذا حدثت مشكلة وأخبرتك رسالة الخطأ أن قاعدة البيانات موجودة من قبل، فافتح مدير سيكويل سيرفر SQL Server Management Studio، وقم بفصل Detach قاعدة بيانات الكتب منه، ثم أعد محاولة الاتصال بها من دوت نت.. ويمكنك أن تلجأ إلى هذه الخطوة كلما واجهت مشكلة في الاتصال بقاعدة البيانات.. فإذا لم يحل هذا المشكلة، فيمكنك عمل Restart لخادم سيكويل نفسه بالطريقة التي شرحناها من قبل.

- بعد نجاح تجربة الاتصال، اضغط OK لإغلاق النافذة.. ستجد أن اسم قاعدة البيانات قد ظهر في متصفح الخوادم، ويمكنك التعامل معها بنفس الطريقة التي اتبعناها في مدير سيكويل.

## إنشاء الفهارس Indices:

نريد الآن إنشاء فهرس لأحد الأعمدة في قاعدة بيانات سيكويل.. من المتوقع في قاعدة بيانات الكتب مثلا، أن يتم البحث عن اسم الكتاب أو اسم المؤلف.. هذا يعني أننا نحتاج إلى فهرسة العمود Authors.Author والعمود Books.Book.. دعنا نرى كيف نفعل هذا:

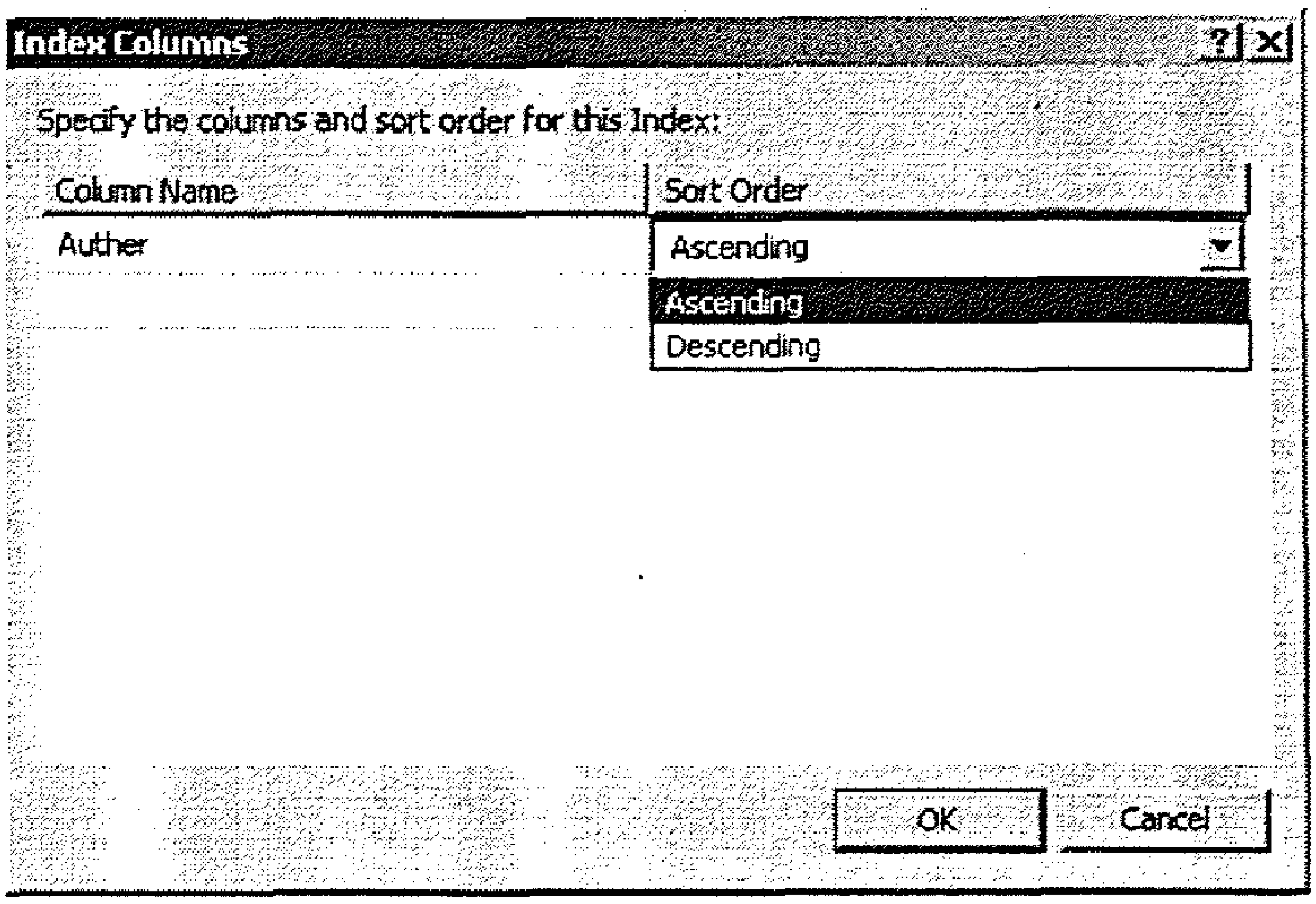
في متصفح الخوادم، انقر مرتين بالفأرة على اسم جدول المؤلفين Authors لفتح تصميم الجدول، واضغط بزر الفأرة الأيمن في أي موضع خال من الجزء العلوي من صفحة التصميم، ومن القائمة الموضعية اختر الأمر Indexes/Keys.. سيعرض هذا نافذة إنشاء الفهارس والمفاتيح الموضحة في الصورة:




كما ترى، ستجد تعريفا للمفتاح الأساسي لهذا الجدول واسمه الافتراضي PK_Authors، وكما ترى من خصائص هذا الفهرس (المفتاح) أنه مرتبط بالحقل ID ومرتب تصاعديا ASC.

والآن، اضغط الزر Add لإضافة فهرس جديد.. سيضاف عنصر إلى الشجرة الموجودة على اليسار اسمه الافتراضي IX_Authors.. لاحظ أن البادئة IX هي اختصار Index، ومن المنطقي افتراض أن العنصر الجديد فهرس وليس مفتاحا أساسيا، لأن الجدول لا يمكن أن يحتوي على أكثر من مفتاح أساسي واحد، بينما يمكن أن يحتوي على أكثر من فهرس.. على كل حال، تستطيع تغيير اسم هذا الفهرس لو أردت، والأفضل أن توضح فيه اسم الحقل أيضا.. غير قيمة الخاصية Name إلى IX_Authors_Author.

علينا الآن أن نحدد الحقل الذي يخصه هذا الفهرس.. اضغط الزر الموجود في خانة قيمة الخاصية Columns.. سيعرض هذا نافذة اختيار أعمدة الفهرس، وهي كما في الصورة:




وظيفة هذه النافذة، هي إتاحة الفرصة لك لاختيار أكثر من عمود لتتم فهرستها معا.. وكما ذكرنا من قبل، سيتم الترتيب على أساس العمود الأول، فإن تشابهت بعض عناصره، فسيتم ترتيبها على أساس العمود الثاني، فإن تشابهت بعض الأزواج من العمودين الأول والثاني معا، فسيتم ترتيبها على أساس العمود الثالث... وهكذا.. كل ما عليك في هذه الحالة، هو الضغط بالفأرة في صف فارغ في النافذة، واختيار اسم الحقل من القائمة المنسدلة من العمود الأول، واختيار نوع الترتيب (تصاعديا Ascending أم تنازليا Desinding) من العمود الثاني.. لكن في معظم الأحول لن تحتاج إلا إلى فهرسة عمود واحد كما في حالتنا هذه.. لهذا اختر اسم العمود Authors واترك الترتيب تصاعديا كما هو وأغلق النافذة بضغط الزر OK للعودة إلى نافذة خصائص الفهرس.

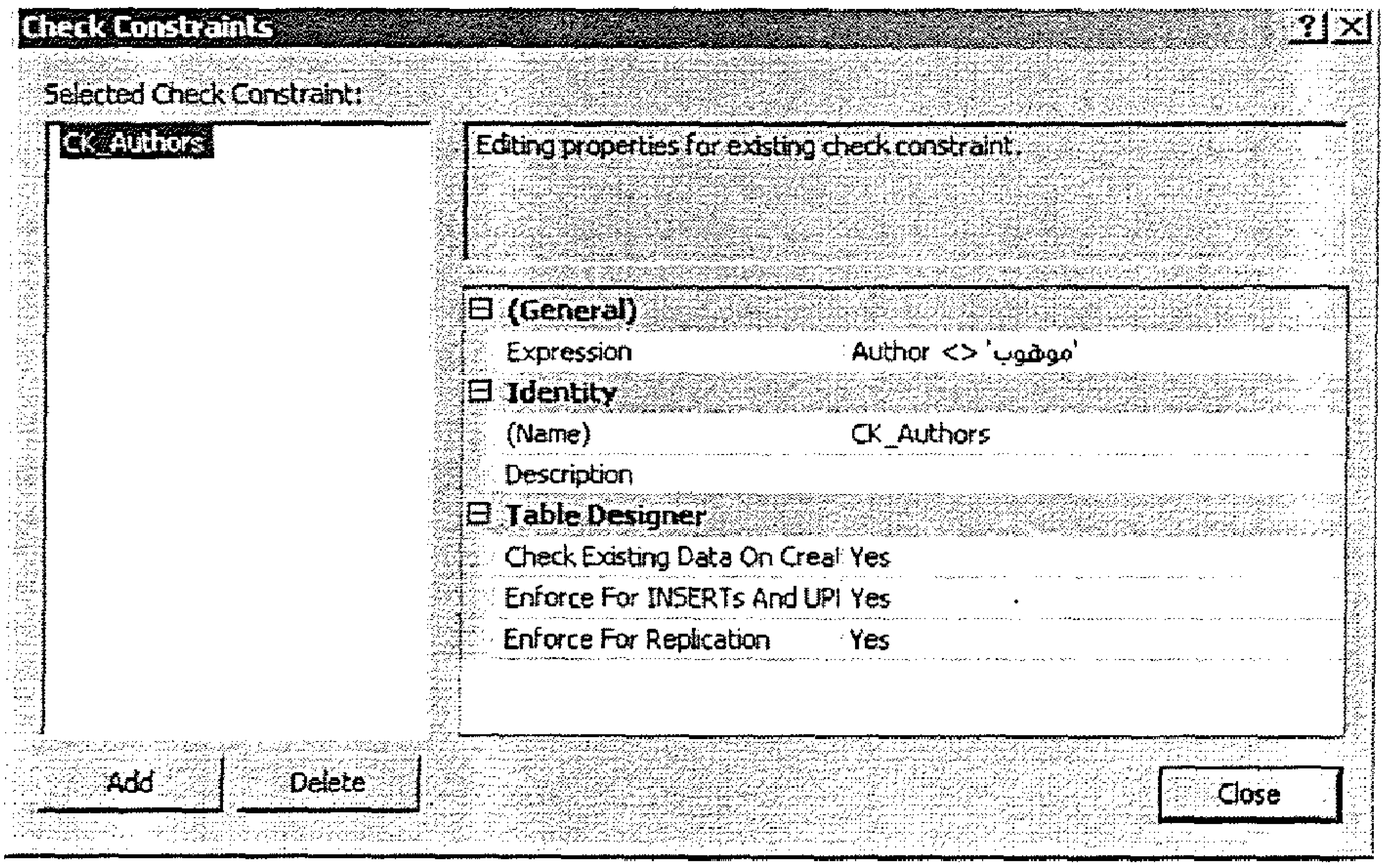
يمكنك تغيير بعض خصائص الفهرس على حسب احتياجك.. مثلا: لو أردت عدم السماح للمستخدم بتكرار أسماء المؤلفين في العمود Author، فاجعل للخاصية Is Unique القيمة Yes.. وإذا أردت اعتبار هذا الحقل مفتاحا متفردا UniqueKey للجدول، فاجعل للخاصية Type القيمة Unique Key بدلا من Index.. وهكذا.

اضغط Close لإغلاق النافذة، واضغط زر الحفظ لحفظ التغييرات التي حدثت للجدول في قاعدة البيانات.. لقد تم إنشاء فهرس حقل المؤلفين الآن!

وبنفس الطريقة، يمكنك إنشاء فهرس لحقل Book في جدول الكتب.

## إنشاء القيود Constraints:

رأينا كيف ننشئ قواعد التحقق من الصحّة في Access.. هنا أيضا يمكن أن نفعل المثل.. اضغط بزر الفأرة الأيمن في أي موضع خال من الجزء العلوي من صفحة تصميم الجدول Authors، ومن القائمة الموضعية اضغط الأمر Check Constrains.. سيؤدي هذا إلى عرض نافذة إنشاء قواعد التحقق من الصحة، كما هي في الصورة:




في الخاصية Expression اكتب قاعدة التحقق من الصحّة.. لاحظ أنّ هناك اختلافا طفيفا عن طريقة كتابة القاعدة في Access، فأنت هنا مضطرّ إلى إدخال اسم الحقل في التعبير، فبينما كانت قاعدة التحقق من الصحّة خاصيّة للحقل في Access، فإنّها هنا جزء من الجدول ككل.. هذا يمنحك القدرة على كتابة قاعدة معقدة تتحقق من قيمة أكثر من حقل معا.. فمثلا، لاستبعاد صديقنا (موهوب) من حقل المؤلفين، اكتب صيغة الشرط كالتالي:

**Author <> 'موهوب'**

ولو كانت صيغة الشرط طويلة جدا، فيمكنك ضغط الزر الموجود في خانة الشرط، لعرض نافذة بها مربع نص متعدد الأسطر، لتكتب شروطا طويلة ومعقدة، وعند ضغط الزر OK لإغلاق نافذة الشرط، سيتم التأكد إن كانت صيغة الشرط الذي كتبته مقبولة أم لا.

أغلق النافذة وقم بحفظ التغييرات، ثمّ انتقل إلى تخطيط الجدول وجرّب كتابة اسم المؤلف الفذ (موهوب)!

***

الآن، نكون قد ألممنا بأهمّ مفاهيم قواعد بيانات سيكيول سيرفر وكيفيّة إنشائها والاتصال بها من دوت نت.. وفي الفصل التالي، سنتعرّف على كيفيّة التعامل معها، باستخدام لغة الاستعلام المركبة SQL.. فإلى هناك.

# لغـــــة الاستعـــــلام المـركّبـــــة
# Structured Query Language (SQL)

مهما كان نوع تطبيق قاعدة البيانات الذي تتعامل معه، فإنه يستخدم لغة خاصّـة بقواعد البيانات متفقـا عليهـا دوليّـا اسـمها SQL (تنطـق "اس كيـو إل" أو "سيكويل").. صحيح أن هناك اختلافات في تركيب هذه اللغة بينَ تطبيق وآخـر، ولكنها اختلافات طفيفة لا تكفي للادعاء بأنها نسخ مختلفة تمام الاختلاف.
وبالمقارنة بلغات البرمجة المألوفة، تعتبـر SQL لغـة غيـر إجرائيّـة Non-procedural، فهي لا تحتوي على تركيبات لغويةٍ مثل جمـل الـشرط وجمـل التكرار وما شابهها.. وتعتبر SQL لغة برمجة عالية المستوي، حيث يمكنها في سطر واحد إجراء عمليّات بليغة التعقيد على قاعدة البيانات.

| **ملحوظة:** |
| --- |
| يقدم لنا SQL Server لغة إجرائية تسمى T-SQL وهي تستخدم جمل SQL مع بعض جمل تعريف المتغيرات وجمل الشرط لكتابة ما يسمى بـالإجراءات المخزنة Stored Procedures كما سنرى لاحقـا.. لكـن نظـرا لأن لغـة T-SQL أكبر من أن نغطيها هنا، فسنتركها إلى كتاب لاحق بإذن الله. |

وتنقسم جمل لغة SQL إلى طائفتين رئيسيّتين:

١- **لغة التعامل مع البيانات Data Manipulation Language (DML):**
وتختصّ باسترجاع أو تحديث أو إضافة أو حذف السجلات التـي تحقـق شروطا معيّنة، لهذا تـسمّى بالاسـتعلامات Queries.. وتـسمّى جمـل استرجاع البيانات باستعلامات التحديد Selection Queries، بينما تسمّى جمل الحذف والتعديل والإضافة باستعلامات الفعل Action Queries.

٢- **لغة تعريف البيانات Data Definition Language (DDL):**
وتختصّ بإنشاء أو حذف كائنات قاعـدة البيانـات، كالجـداول والأعمـدة والفهارس والعلاقات والقيود Constrains.

## باني الاستعلام Query Builder:

لكيّ تجرّب الأمثلة التي سنشرحها في هذا الفصل، يمكنك استخدام نافذة تسمى "باني الاستعلام" Query Builder، لإنشاء جمل SQL بطريقة مرئيّة Visual.

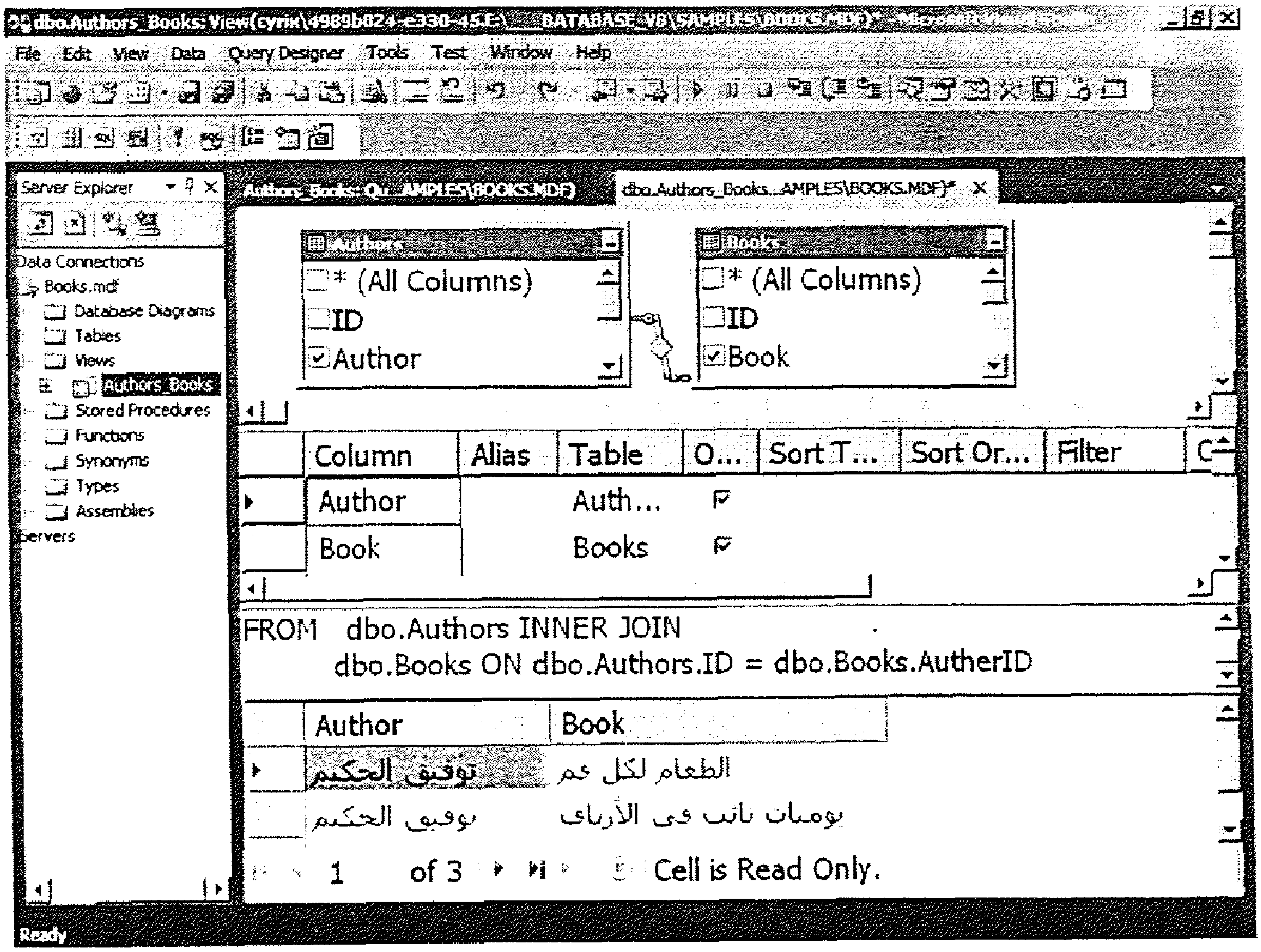
ولعرض باني الاستعلام، أسدل عناصر قاعدة البيانات Books في متصفح خوادم الإنترنت بزرّ الفأرة الأيمن.. ستجد ضمن عناصر قاعدة البيانات عنصرا اسمه Views.. تحت هذا العنصر يمكنك أن تنشئ العُروض Views.

اضغط العنصر Views بزر الفأرة الأيمن، ومن القائمة الموضعيّة اضغط الأمر Add New View.. ستظهر لك نافذة "إضافة جدول" Add Table، وهي تحتوي على أسماء جداول قاعدة بيانات الكتب.. حدّد الجدولين Authors و Books واضغط الزرّ Add، ثم الزر Close.

الآن أنت ترى نافذة باني الاستعلام، التي تتكوّن من أربعة أجزاء 4 Panes:

- جزء المخطط Diagram Pane.
- جزء المعايير Criteria Pane.
- جزء جملة الاستعلام SQL Pane.
- جزء النتائج Results Pane.

وتستطيع عرض أو إخفاء أي جزء من هذه الأجزاء، وذلك بضغط باني الاستعلام بزر الفأرة الأيمن، وضغط اسم الجزء المطلوب من القائمة الفرعية Pane.



### قسم المخطط Diagram Pane:

في هذا الجزء، تظهر الجداول التي اخترت استخدامها في بناء الاستعلام، والتي ستحصل على البيانات منها.. هذه الجداول تظهر في صورة مشابهة للتي رأيناها في مخططات قاعدة البيانات Database Diagrams.

ويمكنك حذف أيّ جدول بضغطه بالفأرة، وضغط الزرّ Delete من لوحة المفاتيح.. ولإضافة الجدول مرّة أخرى، اضغط بزرّ الفأرة الأيمن في أيّ موضع من المخطّط، ومن القائمة الموضعيّة اضغط Add Table، لتظهر لك نافذة إضافة جدول.

وإذا كانت هناك علاقات قد تمّ إنشاؤها بين الجداول سابقا، فستجد الخطوط التي توضّحها مرسومة بين الجداول.. كما يمكنك إنشاء العلاقات بين الجداول، بسحب اسم الحقل من الجدول الرئيسيّ وإسقاطه على اسم الحقل في الجدول الفرعي، بنفس الطريقة التي تعلمناها عند التعامل مع مخططات قواعد البيانات.

وفي حالتنا هذه سيظهر خط يصل بين الحقل ID في جدول المؤلفين، والحقل AuthorID في جدول الكتب، وعلى منتصف هذا الخط سيظهر رمز يمثل نوع الربط الذي سيتمّ تنفيذه في جملة SQL ما بين الجدولين.. لاحظ أن للربط أنواعا عديدة، يمكن اختيارها بضغط الرمز بزرّ الفأرة الأيمن لعرض القائمة الموضعيّة. ويمثل شكل الجوهرة ربطا داخليّا Inner Join، وهذه هي القيمة الافتراضيّة ما لم تختر أنت غير ذلك.. (سنفهم معنى هذه الارتباطات لاحقا).

أول خطوة لبناء جملة SQL، هي اختيار الحقول التي ستظهر في النتيجة.. لفعل هذا اضغط بالفأرة مربّع الاختيار على يسار كل حقل تريد إضافته.. فإذا كنت تريد إضافة كل حقول الجدول مرّة واحدة، فاضغط مربّع اختيار العنصر (All Columns).. وفي مثالنا هذا، اختر الحقل Author من جدول المؤلفين، والحقل Book من جدول الكتب، لنعرض أسماء المؤلفين وكتب كل منهم.

### قسم المعايير Criteria Pane:

يعرض هذا الجزء جدولا يحتوي على أسماء الحقول التي تمّ اختيارها في جزء المخطّط، ويتيح لك تحديد الشروط والمعايير التي تريد تطبيقها على هذه الحقول، لاختيار بيانات معينة منها.

ويمكنك أن توضح أن بعض هذه الحقول تستخدم فقط لقراءة البيانات منها، لكن دونَ أن تظهر في النتيجة.. لفعل هذا أزل العلامة (✓) من خانة العمود Output المناظرة لهذه الحقول.

وتمثل خانة "الاسم المستعار" Alias الاسم الجديد الذي تريد عرض العمود به في النتيجة.. فقد لا تريد مثلا، أن تعرض لمستخدم برنامجك أسماء المؤلفين العرب

تحت عمود اسمه Author، وبدلا من هذا تريد أن تمنح لهذا العمود الاسم المستعار "المؤلف".. أيضا، يمكنك أن تعرض الحقل Book بالاسم "الكتاب".
وهناك حالة أخرى تحتاج فيها لاستخدام الاسم المستعار، تلك إذا كانت النتيجة تحتوي على عمودين أو أكثر من جداول مختلفة لها نفس الاسم، فتحتاج إلى التفريق بينها قبل عرضهما للمستخدم.
ويمكنك ترك الخانة Alias فارغة إذا لم تكن تريد استخدام اسم مستعار.
وسنتعرف لاحقا على كيفية استخدام جزء المعايير Criteria Pane لإضافة الشروط المطلوب تطبيقها على الحقول.

**قسم الاستعلام SQL Pane:**

في هذا الجزء يظهر نصّ جملة SQL، المولد آليا نتيجة اختياراتك التي أجريتها في قسم المخطط وقسم المعايير.. هذا يجعلك بقليل من الملاحظة تتعلم لغة SQL، وذلك بتجريب بعض التغييرات في المخطط وجدول الحقول وملاحظة تأثيرها على جملة SQL.
وبإمكانك التعديل في جملة SQL المكتوبة في هذا القسم على حسب ما يناسبك، وسيظهر أثر هذا التعديل في جزء المخطط وجزء المعايير.. وبإمكانك أيضا كتابة أيّ جملة SQL (أو لصقها) في هذا القسم لتجريبها، كما يمكنك أن تنسخ جملة SQL من هنا لتستخدمها في أيّ موضع آخر من برنامجك.
ولكن.. إلى الآن لم يتمّ تنفيذ الاستعلام، ولم نختبر ناتجه!
اضغط بزرّ الفأرة الأيمن في أيّ موضع من باني الاستعلام، ومن القائمة الموضعيّة اضغط الأمر Run Query، أو اضغط الاختصار Ctrl+R من لوحة المفاتيح مباشرة.

**قسم النتائج Results Pane:**

هذا هو الموضع الذي تظهر به نتائج تنفيذ الاستعلام.. ولمحو النتائج، اضغط قسم النتائج بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اختر الأمر Clear Results.

بعد أن تنتهي من تصميم عرض البيانات View، احفظه بالاسم Authors_Books، ثم أغلق نافذة باني الاستعلام.. الآن سيظهر الاسم Authors_Books تحت العنصر Views في متصفّح الخوادم، ولو أردت إعادة تصميم هذا العرض، فاضغطه بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اضغط الأمر Design View، أو اضغط اسم العرض مرتين بالفأرة مباشرة.. سيعرض هذا باني الاستعلام وفيه تفاصيل الاستعلام الخاص بهذا العرض.

ويمكنك أيضا أن تضغط اسم العرض Authors_Books بزر الفأرة الأيمن، واختيار الأمر Show Results من القائمة الموضعية.. سيعرض هذا جدولا يحتوي على نتائج تنفيذ الاستعلام.

كانت هذه فكرةً سريعة عن باني الاستعلام، ليمكنك استخدامه في تجربة جمل SQL التي سنشرحها في المقاطع التالية.. كل ما عليك هو نسخ الجملة إلى قسم SQL وضغط Ctrl+R لتنفيذها.. وسنتعرّف على باقي إمكانيّات باني الاستعلام أثناء شرحنا لجمل SQL، حيث سننوّه إلى كيفيّة استخدامه لبناء كل جملة.

لاحظ أنك لا تحتاج إلى إنشاء عرض View جديد لكل جملة استعلام تريد تجربتها، فبإمكانك ضغط العنصر Views بزر الفأرة الأيمن، وضغط الأمر "استعلام جديد" New Query، فهذا سيؤدي إلى عرض نافذة باني الاستعلام لتجرب فيها ما تشاء، وبعد هذا تغلقها دون حفظ التغييرات.

والآن، دعنا نتعرف على لغة الاستعلام المركبة SQL.

## استعلامات التحديد Selection Queries

تستخدم هذه الاستعلامات للحصول على بعض البيانات من قاعدة البيانات أو إضافة بيانات جديدة إليها، أو تعديل أو حذف بيانات موجودة فيها.. وفيما يلي نتعرف على أهم جمل هذا النوع من الاستعلامات:

### جملة التحديد SELECT Statement:

تمكنك الجملة SELECT من استرجاع جزء من صفوف الجدول.. وأبسط صورة لهذه الجملة هي:

**SELECT أسماء الأعمدة**
**FROM أسماء الجداول**

لاحظ ما يلي:

١- جمل SQL غير حساسة لحالة الأحرف، لكن جرى العرف على كتابة كلماتها المفتاحية كاملة بحروف كبيرة Capital على سبيل التمييز، لهذا لو كتبت select فلن تختلف عن Select ولا عن SELECT في شيء.

٢- أسماء الأعمدة والجداول يفصل بينها العلامة ",".. مثال:

**SELECT Author, Book**
**FROM Authors, Books**

لاح أنك لو جرّبت هذا المثال فستفاجأ بنتائج غريبة، حيث سيتم تكرار كل سجل من سجلات المؤلفين مع كل أسماء الكتب، ممّا يبدو معه أن كل مؤلف قد ألف كل الكتب!.. هذا خطأ متوقع، لأننا لم نضع أيّ شرط يوضح طريقة الربط بين حقول الجدولين، وهو ما سنتعلمه بعد قليل.

٣- إذا كان اسم الجدول أو اسم الحقل يحتوي على أيّة رموز غير مقبولة برمجيًا (مثل المسافات أو العلامات ":" أو "+" أو "=" ... إلخ) فيجب وضعه بين قوسين مضلعين [ ].. وعامة، يمكنك وضع اسم أي جدول أو اسم أي حقل بين قوسين مضلعين.. مثال:

**SELECT [Author], [Book]**
**FROM [Authors], [Books]**

٤- إذا تشابهت بعض أسماء الحقول في أكثر من جدول، فيجب التمييز بينها بكتابة اسم كل حقل على الصيغة: (اسم الجدول.اسم الحقل).. وعموما، يمكنك كتابة اسم الجدول قبل اسم أي حقل حتى ولو لم يكن له شبيه.. مثال:

**SELECT [Authors].[Author], [Books].[Book]**
**FROM [Authors], [Books]**

ويمكنك إزالة الأقواس المضلعة إذا لم تكن هناك حاجة إليها:

**SELECT Authors.Author, Books.Book**
**FROM Authors, Books**

٥- ستظهر الحقول في السجلات المعادة بالترتيب الذي كتبت أسماءها به في جملة SELECT.. هذا يعني أن الحقل Author سيظهر أولا ثم الحقل Book في نتيجة الاستعلام السابق.

٦- يمكنك كتابة جملة SQL على أكثر من سطر، فهي ليست كأوامر فيجيوال بيزيك تنتهي بنهاية السطر، ولكنها كأوامر ++C تنتهي بالفاصلة المنقوطة "؛"، وإن كان من الممكن عدم كتابة الفاصلة المنقوطة كاختصار.. هذا معناه أن الجملة التالية صحيحة:

**SELECT Author, Book FROM Authors, Books**

لكن تقسيمها على سطرين يجعل قراءتها أسهل.. هذا مجرد تنسيق.

٧- معظم الأخطاء التي تحدث في كتابة جملة SQL تنتج عن الخطإ في كتابة أسماء الحقول والجداول، فانتبه لذلك جيّدا.

٨- لتحديد كل سجلات الجدول، يمكنك كتابة أسمائها جميعا والفصل بينها بالفاصلة.. ولكن هذا سيكون مأساويا لو كان الجدول يحتوي على عشرين عمودا مثلا!.. لهذا تمنحك SQL تسهيلا رائعا، وهو استخدام العلامة "*" (أو الكلمة ALL) للدلالة على تحديد كل أعمدة الجدول كالتالي:

**SELECT ***
**FROM Authors**

أو:

**SELECT All**
**FROM Authors**

ولو أردت استرجاع كل الحقول من جدولين، فاستخدم جملة كالتالية:

**SELECT Authors.*, Books.***
**FROM Authors, Books**

٩- يمكنك كتابة اسم مختصر للجدول بعد اسمه الحقيقيّ مباشرةً، واستخدامه في باقي جملة الاستعلام للتسهيل.. مثال:

**SELECT A.*, B.ID, B.Book**
**FROM Authors A, Books B**

كما يمكنك استخدام الكلمة AS بين الاسم الحقيقي والاسم المختصر كالتالي:

**SELECT A.*, B.ID, B.Book**
**FROM Authors As A, Books As B**

## الفقرة "حيثُ" WHERE Clause:

لم تخترع قواعد البيانات لكي تقرأ منها كل سجلات الجدول كما أدخلتها، فالأهمّ من هذا هو قدرتك على حساب بعض النتائج واستخلاص المعلومات من هذه السجلات.. هنا يبرز الدور الحيويّ للفقرة WHERE، فهي تسمح لك بتحديد الشروط التي سيتمّ على أساسها استرجاع السجلات، بحيث تحصل فقط على السجلات التي تحقق هذه الشروط.

ولن تكون صيغة هذه الفقرة صعبة عليك، فصيغة الشرط الخاصة بها مماثلة للشروط التي تعوّدت كتابتها في جملة If في لغة فيجيوال بيزيك.

والمثال التالي يسترجع كل الكتب التي يحمل مؤلفها الرقم ١ أو ٣:

```
SELECT Book
FROM Books
WHERE AuthorID = 1 OR AuthorID = 3
```

ولكن من هو المؤلف رقم ١ ومن هو المؤلف رقم ٣؟

هذه الطريقة تبدو عديمة الجدوى، فهي غير عمليّة في البرمجة!.. إننا نريد أن نعرف الكتب التي ألفها (توفيق الحكيم) مباشرة دونَ أن نبحث عن رقمه أولا.. لا بأس إذن فهناك حل.. دعنا نستخدم الجملة التالية:

```
SELECT Book
FROM Books, Authors
WHERE Author = 'توفيق الحكيم' AND AuthorID = Authors.ID
```

لاحظ أننا كتبنا اسم جدول المؤلفين في الفقرة FROM رغم أن أيا من حقوله لن يظهر في النتيجة، وذلك لأننا نستخدم حقول هذا الجدول في الفقرة WHERE.

والآن هذا ما سيحدث: سيتمّ اختيار السجلات التي تحمل اسم (توفيق الحكيم) في جدول المؤلفين، ثمّ اختيار السجلات من جدول الكتب التي يتساوى فيها رقم المؤلف مع رقم المؤلف في السجلات المختارة من جدول المؤلفين.. ونظرا لأنه لن يوجد سوى رقم (توفيق الحكيم) فحسب، فإن كتبه فقط هي التي سيتمّ عرضها.

ومن الممكن تبديل ترتيب جملتي الشرط فالترتيب ليس مهمّا، حيث تقوم قاعدة البيانات بتنفيذ الشروط على حسب أولويّة تنفيذها، وليس على حسب ترتيب كتابتها:

```
SELECT Book
FROM Books, Authors
WHERE AuthorID = Authors.ID AND Author = 'توفيق الحكيم'
```

والآن تعال نطوّر المثال السابق، لنحصل على كل الكتب التي ألفها (توفيق الحكيم) أو (عباس العقاد).. في هذه الحالة لا بدّ من عرض اسم المؤلف في النتيجة، وإلا لتعذر معرفة مؤلّف كل كتاب:

```
SELECT Book, Author
FROM Books, Authors
WHERE (Author = 'توفيق الحكيم' OR Author = 'عباس العقاد')
      AND AuthorID = Authors.ID
```

ولكن ماذا لو أردنا أن نعرض كل أسماء الكتب مع ما يناظرها من مؤلفين؟
سيكون الأمر في منتهى البساطة، فسيقتصر الشرط في هذه الحالة على تساوي رقم المؤلف في الجدولين:

```
SELECT Book, Author
FROM Books, Authors
WHERE AuthorID = Authors.ID
```

| ملحوظة: |
|---|
| يمكنك استرجاع جدول فارغ بوضع شرط خاطئ أو مستحيل في الفقرة WHERE (مثل ١ = ٢) كالتالي:<br>SELECT *<br>FROM Books<br>WHERE 1 = 2<br>يفيد هذا عندما تريد أن تحصل على سجل فارغ يحمل أسماء حقول الجدولة كاملة بدون بيانات، بحيث تسمح للمستخدم بإدخال سجلات جديدة، لكن دون العبث بالسجلات القديمة. |

## إنشاء الفقرة WHERE بباني الاستعلام:

لكي تضيف شروطا في باني الاستعلام على الحقول التي تريد استرجاعها، حدد الحقل في جزء المخطط لإضافته إلى جزء المعايير Criteria Pane.. دعنا نجرب مثلا على الحقل ID في جدول المؤلفين.

ستجد في جدول المعايير عمودا اسمه Filter.. هذا هو العمود الذي تستطيع أن تكتب فيه الشرط الذي تريده، لكن بدون كتابة اسم الحقل.. مثلا، لكي نحصل على كتب المؤلف رقم ٥، اكتب في الخانة Filter في صف الحقل ID الشرط:

= 5

ولو أردت المؤلفين الذين يزيد رقمهم عن ٥، فاكتب الشرط:

> 5

وهكذا....

وإذا أردت أن تكتب شرطا آخر مرتبطا بالشرط الأول بالمعامل OR، فانتقل إلى العمود التالي الذي يحمل العنوان OR، واكتب الشرط الذي تريده في هذا

العمود.. فإذا كان هناك المزيد من الشروط، فلديك المزيد من الأعمدة التي تحمل العنوان OR.. فمثلا، لو أردت أن تحصل على المؤلفين الذين أرقامهم ٧ و ١٢ و ٢٠، فاجعل صف الحقل ID في جزء المعايير يبدو كالتالي (سأقتصر على ذكر الأعمدة التي تهمنا فقط):

| Column | Table | Filter | Or | Or |
|---|---|---|---|---|
| ID | Authors | = 7 | = 12 | = 20 |

ولكن ماذا لو أردت أن تكتب شرطا يستخدم المعامل AND؟

في هذه الحالة يجب أن يظهر هذا الشرط في صفّ جديد.. فمثلا، لو وضعت في العمود Filter شرطا على رقم المؤلف ID وشرطا على اسم المؤلف Author، فسيتم ربط الشرطين معا بالمعامل AND.

ولكن.. ماذا لو أردنا أن يظهر نفس الحقل في أكثر من شرط يربطها المعامل AND، كأن نستعلم عن كل المؤلفين ما عدا المؤلفين رقما ٨ و ٩؟

في هذه الحالة يجب أن نكرر اسم الحقل ID في جزء المعايير.. لفعل هذا اضغط بالفأرة في أيّ خانة فارغة في العمود المسمّى Column، ومن القائمة المنسدلة اختر الحقل Authors.ID.

الآن صار لديك صفان للعمود ID.. اكتب في العمود Filter للأول الشرط:

<> 8

واكتب في العمود Filter الخاص بالثاني الشرط:

<> 9

ولكن ماذا لو كان هناك شرطان كل منهما على حقل مختلف، ويربطهما المعامل OR، كأن تريد اختيار المؤلف الذي رقمه ٦ أو اسمه توفيق الحكيم؟

في هذه الحالة اكتب في العمود Filter الخاص بالحقل ID الشرط:

= 6

واترك العمود Filter الخاص بالحقل Author فارغا، واكتب في العمود OR الخاص به الشرط:

= 'توفيق الحكيم'

**الخلاصة:**

الشروط التي تظهر في العمود Filter في صفوف مختلفة يربطها المعامل AND، بينما الشروط التي تظهر في العمود OR يربطها المعامل OR.. ولو أردت ربط حقلين بالمعامل Or فاترك العمود Filter الخاص بأحدهما فارغا، واكتب الشرط الخاص به في العمود OR.

## معاملات المقارنة:

الجدول التالي يلخص المعاملات التي يمكن استخدامها لتكوين شروط مركبة مع الفقرة WHERE:

| المعاملات المنطقية | |
|---|---|
| AND | و |
| OR | أو |
| NOT | ليس |
| **معاملات المقارنة** | |
| = | يساوي |
| <> | لا يساوي |
| > | أكبر من |
| < | أصغر من |
| >= | أكبر من أو يساوي |
| <= | أصغر من أو يساوي |

فمثلا للحصول على أول ٨ كتب في جدول الكتب:

```
SELECT ID, Book
FROM Books
WHERE ID < 9
```

وللحصول على الكتب التي تسبق كتاب "حائرة في الحب" في الترتيب الهجائيّ، استخدم الجملة التالية:

```
SELECT ID, Book
FROM Books
WHERE Book < 'حائرة في الحب'
```

## المعامل "بين" BETWEEN Operator:

يسمح لك هذا المعامل بتحديد المجال الذي ينتمي إليه الحقل.. فمثلا، يمكننا استخدام الجملة التالية للحصول على الكتب التي ينحصر أسعارها بين ٣ و ١٠ جنيهات:

```
SELECT Book, Price
FROM Books
WHERE Price BETWEEN 3 AND 10
```

كما يمكننا استخدام NOT قبل هذا المعامل للحصول على قيم الحقل التي لا تنتمي إلى المجال المحدّد.. والجملة التالية تعيد الكتب التي لا تنحصر أسعارها بين ٣ و ١٠ جنيهات:

```
SELECT Book, Price
FROM Books
WHERE Price NOT BETWEEN 3 AND 10
```

## المعامل "في" IN Operator:

أحيانا لا يحل المعامل BETWEEN كل مشاكلنا، فماذا لو أردنا أن نختار قيما متفرّقة للحقل؟.. في هذه الحالة سيؤدي استخدام المعامل AND إلى كتابة جملة طويلة جدا.

هنا تبرز أهمية المعامل IN، فبعده توضع كل القيم بين قوسين مفصولة بالعلامة , .. والجملة التالية تعيد إليك الكتب التي تحمل الأرقام ٢ و ٦ و ٧ و ١٠:

```
SELECT ID, Book
FROM Books
WHERE ID IN (2, 6, 7, 10)
```

ويمكن نفي الجملة السابقة للحصول على باقي الكتب (التي لا تحمل الأرقام المذكورة) كالتالي:

```
SELECT ID, Book
FROM Books
WHERE ID NOT IN (2, 6, 7, 10)
```

ولا يقتصر الأمر على الأرقام، فالجملة التالية تعيد كل الكتب التي ألفها (توفيق الحكيم) و(نبيل فاروق) و(أحمد رجب):

```
SELECT Book, Author
FROM Books, Authors
WHERE AuthorID = Authors.ID
      AND Author IN ('توفيق الحكيم', 'نبيل فاروق' , 'أحمد رجب')
```

## المعامل "يشبه" LIKE Operator:

يستخدم هذا المعامل في الفقرة WHERE بنفس الطريقة التي يستخدمه بها مبرمجو فيجيوال بيزيك في مقارنة النصوص، ولكن مع بعض الاختلافات الطفيفة، ستدركها عند التعرّف على العلامات الخاصّة التي يستخدمها هذا المعامل، وهي:

| الرمز | الوظيفة |
|---|---|
| % | تعبّر عن أيّ عدد من الحروف (يمكن أن يكون هذا العدد صفرا)، مهما كانت هذه الحروف.<br>مثال: استخدم الجملة التالية للحصول على جميع أسماء المؤلفين التي تحتوي على حرف الميم في أيّ موضع:<br>**SELECT Author**<br>**FROM Authors**<br>**WHERE Author LIKE '%م%'**<br>ولو أردت أن تبحث عن المؤلفين الذين يبدأون بحرف الميم، فاستخدم الصيغة 'م%'. |
| _ | تعبّر عن حرف واحد فقط، مهما كان هذا الحرف.<br>مثال: استخدم الجملة التالية للحصول على جميع أسماء المؤلفين التي يكون حرف الميم فيها هو ثالث حرف:<br>**SELECT Author**<br>**FROM Authors**<br>**WHERE Author LIKE ''__م%'**<br>حيث استخدمنا علامتي "_" متتاليتين لتحفظا موضعي حرفين (أيّ حرفين) يليهما الحرف الثالث وهو الميم، يليه العلامة % لتدل على أن أيّ عدد من الحروف مهما كانت يمكن أن يأتي بعد حرف الميم، بما في ذلك أن يكون حرف الميم هو آخر حرف في النصّ ولا تليه أية حروف. |
| # | تعبّر عن رقم منفرد من ٠ إلى ٩. |
| [ ] | تعبّر عن حرفٍ واحدٍ من الحروف الموجودة بين القوسين.. هذه الحروف إمّا أن تُكتب متتالية مثل:<br>**[ACdF]**<br>وإمّا أن تُكتب على صورة مجال، مثل:<br>**[g-y]**<br>حيث تعبّر هذه الصيغة عن الحروف من g إلى y. |

| | |
|---|---|
| | مثال: استخدم الجملة التالية للحصول على جميع أسماء المؤلفين التي يكون حرف الميم أو النون أو الواو فيها هو ثالث حرف:<br>**SELECT Author**<br>**FROM Authors**<br>**WHERE Author LIKE "__[منو]%'**<br>ولو أردت أن تبحث عن المؤلفين الذين تبدأ أسماؤهم بأحد الحروف المحصورة بين الفاء والياء وتنتهي بحرف السين، فاستخدم الصيغة:<br>**'[ف-ي]%س'** |
| [^] | تعبّر عن أيّ حرفٍ غير ذلك الموجود في القوسين (سواء من الحروف المذكورة صراحة أو من الحروف التي تقع في المجال المحدّد)..<br>باختصار:هذا هو نفي الصيغة السابقة.<br>مثال: استخدم الجملة التالية للحصول على جميع أسماء المؤلفين التي لا يكون ثالث حرف فيها A أو B أو C:<br>**SELECT Author**<br>**FROM Authors**<br>**WHERE Author LIKE "__[^ABC]%'**<br>ولو أردت أن تبحث عن السجلات التي لا تبدأ بأحد الحروف المحصورة بين T و Z وتنتهي بالحرف S، فاستخدم الصيغة:<br>'[^T-Z]%S' |

| **ملحوظة:** |
|---|
| إذا أردت البحث في النصّ عن أيّ علامة من هذه العلامات الخاصّة، فضعها بين قوسين مضلعين.. فمثلا، يمكنك استخدام الصيغة '%[%]50%' للبحث عن النصّ "50%" في أيّ موضع من النص. |

## المقارنة بناتج استعلام:

من الإمكانيّات التي تمنحها لك الفقرة WHERE، قدرتك على مقارنة قيمة أحد الحقول، بقيمة أيّ حقل من جدول آخر، ناتج عن جملة SELECT كاملة أخرى! انظر للمثال التالي، وفيه نحصل على كل الكتب التي ألفها (عباس العقاد) و(توفيق الحكيم):

```
SELECT *
From Books
WHERE AuthorID = (SELECT ID
                  FROM Authors
                  WHERE Author = 'عباس العقاد'
                  OR Author = 'توفيق الحكيم')
```

حيث تُعيد SELECT الفرعيّة حقلا به رقمي هذين المؤلفين، لتقوم جملة SELECT الخارجيّة بإيجاد السجلات التي تحتوي على أيّ من هذين الرقمين في الحقل AuthorID.

لاحظ أنك تستطيع اختصار الجملة السابقة إلى ما يلي:

```
SELECT Books.*
From Authors , Books
WHERE AuthorID = Authors.ID
      AND Author = 'عباس العقاد' OR Author = 'توفيق الحكيم'
```

## بعض الدوال المستخدمة في إنشاء الشرط:

يمكنك استخدام بعض الدوال في تكوين شرط WHERE، مثل دول أخذ جزء من النص SUBSTRING و LEFT و RIGHT، ودالة حذف المسافات الطرفية TRIM ودالة طول النص LEN.. والمثال التالي يعيد الناشرين الذين تنتهي أسماؤهم بالحرفين "ية":

```
SELECT Publisher
FROM   Publishers
WHERE (RIGHT(Publisher, 2) = 'ية')
```

أما المثال التالي فيعيد أطوال أسماء الناشرين:

```
SELECT LEN(Publisher)
FROM   Publishers
```

## وضع اسم مستعار باستخدام الكلمة AS:

رأينا من قبل كيف يمكن استخدام الكلمة AS لوضع اسم مستعار للجدول.. مثلا:

```
SELECT A.*, B.ID, B.Book
FROM Authors As A, Books As B
```

ويمكنك أيضا استخدام الكلمة AS لوضع اسم مستعار للحقل.. مثال:

```
SELECT Book AS [اسم الكتاب], Author AS [مؤلفه]
FROM Books, Authors
WHERE AuthorID = Authors.ID
```

الآن ستظهر أسماء المؤلفين وكتبهم، ولكن عنواني العمودين سيكونان اسمين عربيّين هذه المرّة.

ومن الاستخدامات الطريفة للاسم المستعار، استخدامه لتسميّة أحد الحقول الناتجة عن دمج حقلين معا.. افترض أننا نريد عرض حقل يحتوي على اسم الكتاب متبوعا بشرطة متبوعة باسم المؤلف.. لفعل هذا، علينا تشبيك Concatenate الحقلين، تماما كما نفعل مع المتغيّرات النصيّة في لغات البرمجة العادية، مع منح الحقل الناتج اسما مستعارا باستخدام الكلمة AS، كالتالي:

```
SELECT Book + '_' + Author AS [اسم الكتاب]
FROM Books, Authors
WHERE AuthorID = Authors.ID
```

| ملحوظة: |
| --- |
| عند تنفيذ الجملة السابقة في باني الاستعلام سيتم وضع التعبير Book + '_' + Author كحقل في قسم المعايير Criteria Pane، مع منحه الاسم المستعار "اسم الكتاب".. هذه هي الطريقة التي تضيف بها الحقول المدمجة في باني الاستعلام.. اكتب العمليّة التي تجمع بين الحقلين في أيّ خانة فارغة في العمود Column، وامنح هذا الحقل الجديد الاسم المستعار المناسب. |

وليست الحقول النصيّة فقط هي ما نستطيع دمجه، حيث يمكننا أن نجرى عمليّات الجمع والطرح والضرب والقسمة على الحقول الرقميّة كما يحلو لنا.

## الكلمة TOP:

يمكنك استخدام هذه الكلمة بعد SELECT مباشرة، وذلك إذا كان عدد السجلات الناتجة عن تنفيذ الاستعلام كبيرا، بينما ينحصر اهتمامك في مجموعة قليلة منها فقط.. فمثلا، لكي تعرض أول خمس سجلات من أسماء الكتب ومؤلفيها استخدم الجملة التالية:

```
SELECT TOP 5 Book, Author
FROM Books, Authors
WHERE AuthorID = Authors.ID
```

وإذا لم تكن تعرف بالضبط عدد السجلات المطلوبة، فيمكنك استخدام النسبة المئويّة PERCENT كالتالي:

```
SELECT TOP 5 PERCENT Book, Author
FROM Books, Authors
WHERE AuthorID = Authors.ID
```

## الكلمة "منفصل" DISTINCT:

استخدم هذه الكلمة لإزالة السجلات المكرّرة من ناتج الاستعلام.. فمثلا، لو عرضنا جدولا بأسماء الكتب بدون مؤلفيها، فقد يتكرر اسم أحد الكتب أكثر من مرة (لمؤلفين مختلفين).. هنا يمكن تلافي هذا التكرار باستخدام الجملة التالية:

```
SELECT DISTINCT Book
FROM Books
```

## القيم المنعدمة Null Values:

تشيرُ القيمة Null في قاعدة البيانات، إلى أن الخانة لم توضع بها أيّ قيمة من أساسِه.. لهذا تختلف الخانة التي بها نص فارغ "" عن الخانة التي ليس بها أي شيء على الإطلاق.

وعند تنفيذ الفقرة WHERE، فإنها لا تسترجع أيّ خانة فارغة، لأن القيمة NULL تنتج False بطريقة دائمة في أيّ عمليّة مقارنة.. هذا بالإضافة إلى أن دخول هذه القيمة في أيّ عمليّة حسابيّة يؤدّى إلى حدوث خطإ في البرنامج.

لهذا يجب أن تعامل هذه القيمة بحذر، حيث يمكنك أن تختبر وجودها باستخدام أحد التعبيرين IS NULL أو IS NOT NULL.. وكمثال، استخدم الاستعلام التالي للبحث عن أيّ خانة في الحقل Author تمّ تركها فارغة:

```
SELECT * FROM Authors
WHERE Author IS NULL
```

لاحظ أنك تستطيع تلافي الكثير من احتمالات الخطأ الناتجة عـن القيمـة Null، بإزالة علامة الاختيار من العمود Allow Nulls عند تصميم الحقول، بحيـث لا يستطيع مدخل البيانات ترك خانة فارغة في هذا الحقل.

## فقرة الترتيب ORDER BY:

يمكنك ترتيب السجلات الناتجة من الاستعلام تبعا لحقلٍ أو أكثر.. وفـي المثـال التالي سنعرض أسماء الكتب ومؤلفيها مرتبة باسم الكتاب:

```
SELECT Book, Author
FROM Books, Authors
ORDER BY Book
WHERE AuthorID = Authors.ID
```

ولكن لو كانت هناك كتب تحمل نفس الاسم، فكيف سيتمّ ترتيبها؟
في هذه الحالة يمكنك أن تحدّد حقل المؤلفين كمفتاح ثانٍ للترتيـب، بحيـث لـو تشابهت أسماء الكتب، يتم ترتيب الكتب المتشابهة على حسب أسماء مؤلفيها:

```
SELECT Book, Author
FROM Books, Authors
ORDER BY Book, Author
WHERE AuthorID = Authors.ID
```

ويكون هذا الترتيب تصاعديًا في الوضع الافتراضيّ.. ولو أردت أن تغير طريقة الترتيب، فاستخدم الكلمة DESC بعد اسم أيّ حقلٍ تريد ترتيبه تنازليًا، والكلمـة ASC بعد اسم الحقل الذي تريد ترتيبه تصاعديًا (وهي افتراضيّة ويمكـن عـدم كتابتها).. مثال:

```
SELECT Book, Author
FROM Books, Authors
ORDER BY Book DESC, Author ASC
WHERE AuthorID = Authors.ID
```

| ملحوظة: |
|---|
| لوضع كيفيّة الترتيب في باني الاستعلام، حدد الحقل الذي تريد أن يتمّ الترتيب على أساسه في قسم المخطط Diagram Pane واضغطه بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اضغط الأمر Sort Ascending إذا كنت تريد الترتيب تصاعديّا، أو الأمر Sort Descending إذا كنت تريد الترتيب تنازليّا.. ستجد أن أيقونة تمثل نوع الترتيب قد ظهرت بجوار اسم الحقل.. ولو أردت إزالة الترتيب، فاضغط نفس الأمر من القائمة الموضعيّة مرّة أخرى.. ويمكنك فعل ذلك مع أكثر من حقل ومن أيّ جدول، حيث ستظهر كلها في جملة SQL بالترتيب الذي أضفتها به.<br>كما يمكنك اختيار نوع الترتيب بطريقة أخرى، وذلك باستخدام قسم المعايير Criteria Pane، حيث يمكنك استخدام القائمة المنسدلة في العمود الذي يحمل العنوان Sort Type لتغيير طريقة ترتيب أي حقل.<br>وإذا أردت استخدام أكثر من حقل للترتيب على أساسها، فعليك استخدام العمود Sort Order للتحكم في أولوية هذه الحقول في عملية الترتيب، فالحقل الذي سيتم الترتيب على أساسه أولا أعطه الرقم ١، والحقل الذي سيتم الترتيب على أساسه في حالة تشابه قيم الحقل الأول أعطه الرقم ٢، وهكذا. |

## دوال التجميع Aggregate functions:

تمنحك SQL بعض الدوال الجاهزة لحساب بعض النتائج.. لاحظ ما يلي:

- هذه الدوال تقبل معاملا واحدا فقط، هو أحد أعمدة الجدول، أو أيّ عمود جديد ناتج عن إجراء عمليّة حسابيّة (كالجمع والطرح والضرب والقسمة) على واحد أو أكثر من الأعمدة، أو أيّ عمود ناتج من جملة SELECT فرعيّة.
- هذه الدوال تعيد قيمة واحدة فقط (رقما منفردا).. أيّ أن الناتج منها هو عمود يحتوي على خانة واحدة فقط.. ولو كنت ستعرض هذه النتيجة، فاستخدم التعبير AS لمنح هذا العمود الجديد اسما مناسبا، وإلا فإن SQL ستمنحه اسما افتراضيّا من لديها.
- لا مانع من استخدام الفقرة WHERE لتحديد السجلات التي ستتمّ العمليّة الحسابيّة عليها.

وهذه الدوال هي:

### العدد COUNT:

تحسب عدد الخانات في العمود المرسل كمعامل.. ويمكن تطبيقها على أيّ نوع من البيانات.. والجملة التالية تحسب عدد الكتب التي ألفها توفيق الحكيم:

```
SELECT COUNT(Book) AS [عدد الكتب المتاحة]
FROM Books, Authors
WHERE Author = 'توفيق الحكيم'
        AND AuthorID = Authors.ID
```

لاحظ أن الحقول التي تحتوي على القيمة NULL لا يتمّ عدّها ضمنَ السجلات.. ولو أردت أن تفعل العكس، فعليك أن ترسل الرمز (*) كمعامل لهذه الدالة، حتى تأخذ هذه السجلات في اعتبارها:

```
SELECT COUNT(*) AS [عدد المؤلفين]
From Authors
```

### العدد الكبير COUNT_BIG:

مماثلة للدالة COUNT في كل شيء، إلا أن ناتج الدالة COUNT يكون عددا صحيحا من النوع int، بينما ناتج الدالة COUNT_BIG يكون عددا صحيحا كبيرا bigint، لهذا عليك استخدامها عند التعامل مع جداول ضخمة يتجاوز عدد سجلاتها ٢ مليار سجل!

### المجموع SUM:

تحسب مجموع الخانات في العمود المرسل كمعامل.. ويمكن تطبيقها على الأعمدة الرقميّة فقط.

مثال: الجملة التالية تحسب مجموع النسخ المتاحة من جميع الكتب:

```
SELECT SUM(Copies_No) AS [إجمالي النسخ]
From Books
```

### المتوسط AVG:

تحسب المتوسّط الحسابيّ (مجموع العناصر ÷ عددها) لخانات العمود المرسل كمعامل.. ويمكن تطبيقها على الأعمدة الرقميّة فقط.. والمثال التالي يحسب متوسط عدد نسخ الكتب:

```
SELECT AVG(Copies_No) AS [متوسط النسخ]
From Books
```

- **الأصغر MIN:**

تحسب أصغر قيمة في العمود المرسل كمعامل.. ويمكن تطبيقها على الأرقام والنصوص، وفي حالة النصوص ستعيد أصغر نصّ في الترتيب الأبجديّ.

- **الأكبر MAX:**

تحسب أكبر قيمة في العمود المرسل كمعامل.. ويمكن تطبيقها على الأرقام والنصوص، وفي حالة النصوص ستعيد أكبر نصّ في الترتيب الأبجديّ.

- **المجموع التأكيدي CHECKSUM_AGG:**

هذه الدالة خاصية بـ T-SQL فقط ولا تستطيع استخدامها مع قواعد بيانات آكسيس.. وهي تجري عملية حسابية على جميع قيم العمود، وتعيد المجموع التأكيدي Check Sum، وهو قيمة يمكنك استخدامها للتأكد من أن خانات العمود لم يحدث بها تغيير، فطالما ظلت ثابتة فهذا معناه أن قيم العمود لم تتغير.. هذا أفضل من حفظ جميع قيم العمود القديمة، ثم التأكد من أن كلا منها لم تتغير على حدة.. لاحظ أن هناك احتمالا صغيرا في أن تتغير بعض قيم العمود لكن تظل هذه الدالة تعطي نفس الناتج.. لاحظ أيضا أن هذه الدالة لا تهتم بكيفية ترتيب العمود.. مثال:

```
SELECT CHECKSUM_AGG(Copies_No)
FROM   Books
```

وتوجد دالة أخرى في T-SQL اسمها CHECKSUM، ولكنها ليست دالة تجميع، فهي تعيد عمودا جديدا وليس قيمة واحدة فقط.. وتنتج كل خانة في العمود العائد من حساب قيمة مختلطة Hash Value لكل خانة في العمود الأصلي.

- **التغير الإحصائي VAR:**

تحسب التغير الإحصائي Statistical Variance للقيم الموجودة في العمود.. لو لم تدرس مادة الإحصاء من قبل، فلا تشغل بالك بهذه الدالة وكل الدوال التالية!

- **التغير الإحصائي السكاني VARP:**

تحسب التغير الإحصائي السكاني Statistical variance for population للقيم الموجودة في العمود.

### الانحراف الإحصائي المعياري STDEV:

تحسب الانحراف الإحصائي المعياري Statistical Standard Deviation للقيم الموجودة في العمود.

### الانحراف الإحصائي المعياري السكاني STDEVP:

تحسب الانحراف الإحصائي المعياري السكاني Statistical Standard Deviation for Population لقيم العمود.

ويمكنك استخدام القيم المعادة من هذه الدوال في شروط الفقرة WHERE.. والمثال التالي يريك كيف نحسب عدد الكتب التي ألفها أوّل مؤلف في الترتيب الأبجديّ:

```
SELECT COUNT(Book) AS [عدد كتب المؤلف الأول]
FROM Books, Authors
WHERE AuthorID = Authors.ID
    AND Author = (SELECT MIN(Author)
                  From Authors)
```

حيث استخدمنا جملة SELECT فرعيّة لتعيد ناتج الدالة MIN في شرط الفقرة WHERE.. على كل حال، هناك صيغة ثانية للمثال الأخير، باستخدام الربط الداخليّ INNER JOIN الذي سنتعرّف عليه لاحقا.

ولكن ماذا لو أردت أن تطبّق هذه الدوال على حقل به قيم مكرّرة، وأردت ألا تأخذ التكرار في اعتبارك؟

في هذه الحالة يجب أن تستخدم الكلمة DISTINCT للحصول على حقل ليس به أيّ تكرار.. وتوضع هذه الكلمة قبل اسم الحقل مباشرة (داخل قوس الدالة).. والمثال التالي يريك كيف نحسب عدد المؤلفين بحساب عدد خانات الحقل AuthorID في جدول الكتب بدون تكرار:

```
SELECT COUNT (DISTINCT AuthorID) AS [عدد المؤلفين]
From Books
```

| ملحوظة: |
| --- |
| لاستخدام دوال التجميع في باني الاستعلام، اضغط اسم الحقل في قسم المعايير Criteria Pane بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اضغط الأمر Add Group By.. سيظهر عمود جديد في جدول الحقول اسمه Group By، وسيكون مكتوبا فيه مبدئيا النص Group By.. اضغط زرّ إسدال القائمة، حيث ستجد بها أسماء دوال التجميع التي يمكنك استخدامها.. اختر منها ما تريد لتطبيقه على هذا الحقل. |

## تجميع السجلات باستخدام التعبير GROUP BY:

رأينا كيف نجري بعض العمليّات الحسابيّة باستخدام دوال التجميع.. ولكن ماذا لو أردنا مثلا أن نحصل على سجل يحتوي على عدد الكتب التي ألفها كل مؤلف؟
في هذه الحالة لن تسعفنا التعبيرات التي تعلمناها حتى الآن.
هنا تبرز أهمّية التعبير GROUP BY، الذي يقوم بتقسيم سجلات الجدول إلى مجموعات فرعية، تمتلك كل مجموعة منها القيمة نفسها في حقل معين (كرقم المؤلف مثلا)، ومن ثم يتم تطبيق دوال التجميع على كل مجموعة فرعية بمفردها، وبهذا يكون ناتج دوال التجميع عمودا يحتوي على مجموعة من الصفوف، وليس صفا واحدا كما ألفنا من قبل.. هذه هي الجملة التي تحسب عدد كتب كل مؤلف:

```
SELECT AuthorID, COUNT(AuthorID) AS [عدد الكتب]
FROM Books
GROUP BY AuthorID
ORDER BY [عدد الكتب]
```

لاحظ قدرتنا على استخدام الاسم المستعار للعمود الناتج في باقي جملة SQL.
ولكن.. الجدول الناتج من الاستعلام السابق يحتوي على أرقام المؤلفين وليس أسماءهم.. لهذا لا ضير من استخدام الفقرة Where للربط بين الجدولين كالتالي:

```
SELECT Author, COUNT(AuthorID) AS [عدد الكتب]
FROM Authors, Books
Where  AuthorID = Authors.ID
GROUP BY Author
ORDER BY [عدد الكتب]
```

لاحظ أننا قمنا بعملية التجميع في الجملة الأخيرة باستخدام الحقل Author.. السبب في هذا، هو أن هناك قيدا صارما على الحقول المذكورة في المقطع SELECT، وهو أنها جميعا يجب أن تظهر إما في المقطع GROUP BY وإما في المقطع ORDER BY.. لهذا لو تركنا التجميع على الحقل AuthorID فستعترض سيكويل سيرفر على ظهور اسم الحقل Author في المقطع SELECT!

| ملحوظة: |
|---|
| لاستخدام الفقرة Group By في باني الاستعلام، اضغط اسم الحقل في قسم المعايير Criteria Pane بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اضغط الأمر Add Group By.. سيظهر عمود جديد في جدول الحقول اسمه Group By، وسيكون مكتوبا فيه مبدئيا النص Group By.. هذا معناه تجميع القيم المتشابهة لهذا الحقل.<br>ولو أردت إضافة دالة تجميع مع الفقرة Group By لنفس الحقل، فكرر اسم الحقل مرة أخرى في قسم المعايير بضغط صف فارغ بالفأرة واختيار اسم العمود من القائمة المنسدلة الموجودة في العمود Column، وبهذا تستطيع أن تضيف إلى هذا الحقل دالة التجميع التي تريدها. |

## استخدام الفقرة HAVING:

ماذا لو أردنا عرض أسماء المؤلفين الذين تزيد كتبهم عن كتاب واحد؟

في هذه الحالة يمكننا استخدام الفقرة HAVING، وهي فقرة شرطية تسمح باستخدام دوال التجميع في الشرط، كالتالي:

```
SELECT Author, COUNT(AuthorID) AS [عدد الكتب]
FROM Authors, Books
Where  AuthorID = Authors.ID
GROUP BY Authors.Author
HAVING COUNT(AuthorID) > 1
ORDER BY [عدد الكتب]
```

| ملحوظة: |
|---|
| لبناء الفقرة HAVING السابقة في باني الاستعلام.. اذهب إلى الخانة Group By الخاصة بالعمود AuthorID في قسم المعايير، ومن القائمة المنسدلة اختر الدالة Count، ثم انتقل إلى الخانة Filter المجاورة، واكتب فيها الشرط 1<.. هذا معناه أن أي شرط تكتبه في الخانة Filter أثناء ظهور العمود Group By سيظهر كشرط في الفقرة Having.. لكن لو كنت تريد استخدامه كشرط في الفقرة Where، فأسدل القائمة المنسدلة في الخانة Group By واختر منها العنصر Where. |

## عمليات الربط SQL Joins:

تحدّد عمليّات ربط الجداول Joins كيفيّة استعادة البيانات من الجداول التي بينها علاقات.. وهناك خمسة أنواع من عمليّات الربط:

١- الربط المتقاطع Cross Join.
٢- الربط الأيسر Left Join.
٣- الربط الأيمن Right Join.
٤- الربط الكامل Full Join.
٥- الربط الداخليّ Inner Join.

دعنا نتعرف على هذه الانواع.

## الربط المتقاطع Cross Join:

لقد تعرفنا على هذا النوع من الربط سابقا، فهو ينتج تلقائيا عند عرض حقلين من جدولين مختلفين بدون ذكر أي طريقة لربطهما معا.. في هذه الحالة، يتم ما يشبه عملية الضرب، حيث يتم تكوين أزواج تبادلية من الحقلين، بحيث تظهر كل خانة من الجدول الأول مع كل خانة من الجدول الثاني في هذه الأزواج، وبهذا يكـون عدد السجلات الناتجة = عدد خانات الحقل الأول × عدد خانات الحقل الثـاني.. مثال:

```
SELECT Author, Book
FROM   Authors, Books
```

عند تنفيذ هذا الاستعلام، سيظهر ناتج شبيه بما يلي (سنفترض أن هناك مـؤلفين اثنين فقط على سبيل الاختصار):

| Author | Book |
|---|---|
| توفيق الحكيم | الطعام لكل فم |
| توفيق الحكيم | شهرزاد |
| توفيق الحكيم | مهنتي القتل |
| توفيق الحكيم | الاحتلال |
| نبيل فاروق | الطعام لكل فم |
| نبيل فاروق | شهرزاد |
| نبيل فاروق | مهنتي القتل |
| نبيل فاروق | الاحتلال |

طبعا هذا ناتج عجيب وليس عمليا هنا، ولكنه قد يبدو عمليا في بعض التطبيقات الرياضية التي تحتاج فيها إلى الحصول على تباديل قيمتين أو أكثر.
وتوجد فقرة خاصة بهذا النوع من الربط، وهي الفقرة CROSS JOIN، ولو كتبت الاستعلام السابق في باني الاستعلام، فسيتم تحويله تلقائيا إلى الصيغة التالية:

**SELECT Author, Book**
**FROM Authors CROSS JOIN Books**

| **ملحوظة:** |
|---|
| لإنشاء الجملة السابقة باستخدام باني الاستعلام، حدّد الحقل Book من جدول الكتب، والحقل Author من جدول المؤلفين، ثمّ اضغط بزرّ الفأرة الأيمن على الخط الواصل بينَ الجدولين، ومن القائمة الموضعيّة اضغط الأمر: Delete لحذف الربط بين الجدولين.. هذا سيجعل الربط بينهما تقاطعيا!.. لاحظ أن حذف هذا الخط لا يحذف العلاقة الموجودة بين الجدولين، ولكنه فقط يؤثر على كيفية بناء الاستعلام.. ولإعادة رسم الخط مرة أخرى، اسحب الحقل Authors.ID وأسقطه على الحقل Books.AuthorID. |

**<u>الربط الأيسر Left Join:</u>**

تعرض هذه العمليّة كل سجلات الجدول الأيسر (الموجود في بداية الصيغة)، مع بعض سجلات الجدول الأيمن، التي تحقق شرط الربط (الذي يأتي بعد الكلمة ON).
ويمكننا أن نعرض كل أسماء المؤلفين وأسماء الكتب الخاصة بكل منهم، كالتالي:

**SELECT Book, Author**
**FROM Authors LEFT JOIN Books**
**ON AuthorID = Authors.ID**

ولكي تشعر بوجود اختلاف عن ناتج الجملة WHERE، يجب أن يكون هناك بعض المؤلفين الذين لا توجد لهم كتب مناظرة في جدول الكتب، فاستخدام الربط الأيسر سيعرض أسماء كل المؤلفين مع ترك خانة الكتاب فارغة NULL للمؤلف الذي لا توجد له كتب.. لكن دون عرض الكتب التي لا يناظرها مؤلفون.. بينما الجملة WHERE لا تعرض إلا المؤلفين الذي لهم كتب.

| **ملحوظة:** |
|---|
| لإنشاء الجملة السابقة باستخدام باني الاستعلام، حدّد الحقل Book من جدول الكتب، والحقل Author من جدول المؤلفين، ثمّ اضغط بزرّ الفأرة الأيمن على علامة الربط في منتصف الخط الواصل بينَ الجدولين، ومن القائمة الموضعيّة اضغط الأمر: Select All Rows From Authors.. هذا هو كل شيء! |

**الربط الأيمن Right Join:**

مماثل للربط الأيسر، إلا إن كل سجلات الجدول الأيمن يتمّ عرضها بالكامل، مع عرض سجلات الجدول الأيسر التي تحقق شرط الربط.. مثال:

```
SELECT Book, Author
FROM Authors RIGHT JOIN Books
ON AuthorID = Authors.ID
```

ولكي تشعر بتأثير الربط الأيمن، يجب أن يكون هناك بعض الكتب التي لا يوجد لها مؤلف في جدول المؤلفين، فاستخدام الربط الأيمن سيعرض أسماء كل الكتب، والكتاب الذي ليس له مؤلف سيتم ترك خانة المؤلف المناظرة له فارغة NULL. تذكر أننا عندما أنشأنا الحقل AuthorID أزلنا علامة الاختيار من الخاصيّة Allow Nulls لهذا لن يمكنك ترك خانة رقم المؤلف فارغة.. ويمكنك تغيير قيمة هذه الخاصيّة وتجربة إدخال كتاب بدون مؤلف، لترى تأثير عمليّة الربط الأيمن.

| **ملحوظة:** |
| --- |
| لإنشاء الجملة السابقة باستخدام باني الاستعلام، حدّد الحقل Book من جدول الكتب، والحقل Author من جدول المؤلفين، ثمّ اضغط بزرّ الفأرة الأيمن على علامة الربط في منتصف الخط الواصل بينَ الجدولين، ومن القائمة الموضعيّة اضغط الأمر: Select All Rows From Books.. هذا هو كل شيء! |

**الربط الكامل Full Join:**

هذا النوع هو مزيج من الربط الأيمن والأيسر، وفيه يتمّ عرض كل بيانات الجدولين التي تحقق شرط الربط.. مثال:

```
SELECT Book, Author
FROM Authors FULL JOIN Books
ON AuthorID = Authors.ID
```

ولكي تشعر بتأثير الربط الكامل، يجب أن يكون هناك بعض المؤلفين الذين لا توجد لهم كتب، وبعض الكتب التي لا يوجد لها مؤلف، حيث سيتمّ عرض أسماء هؤلاء المؤلفين وهذه الكتب، مع ترك الخانة المناظرة فارغة NULL.

| **ملحوظة:** |
| --- |
| لإنشاء الجملة السابقة باستخدام باني الاستعلام، اتبع نفس الخطوات التي اتبعناها في إنشاء الربط الأيسر، ولكن اختر كلا الاختيارين Select All Rows From Books، وSelect All Rows From Authors.. لاحظ تغير شكل الأيقونة المجاورة عند اختيار أي من هذين الخيارين.. ولإلغاء أي اختيار اضغطه مرة أخرى. |

## الربط الداخليّ Inner Join:

هذه العمليّة مماثلة للفقرة WHERE، حيث يتمّ عرض السجلات المتوافقة فقط من الجدولين، مع تجاهل الخانات الفارغة:

**SELECT Book, Author**
**FROM Authors INNER JOIN Books**
**ON AuthorID = Authors.ID**

| **ملحوظة:** |
|---|
| لإنشاء الجملة السابقة باستخدام باني الاستعلام، حدّد الحقل Book من جدول الكتب، والحقل Author من جدول المؤلفين.. هذا كل شيء! ولو كنت اخترت الربط الأيسر، أو الربط الأيمن، فاضغط رمز العلاقة بزر الفأرة الأيمن، ومن القائمة الموضعية أعد ضغط الأمر Select All Rows From Books، أو Select All Rows From لإزالة اختياره. |

لاحظ أن الربط الأيسر والربط الأيمن والربط الكامل تعتبر جميعا أنواعا من الربط الخارجي Outer Join.. لهذا يمكنك أن تضيف الكلمة OUTER بعد كل من LEFT JOIN و RIGHT JOIN و FULL JOIN، وهو ما يفعله باني الاستعلام تلقائيا، فهو يضيف الكلمة OUTER حتى لو لم تكتبها أنت.

## استعلامات الأداء Action Queries:

يعتبر هذا النوع من الاستعلامات أبسط من استعلامات التحديد، فهو لا يقوم باسترجاع أيّ سجلات.. ولكنه في المقابل يغير بعض بيانات الجدول، سواء بتحديث قيم السجلات أو بإضافة سجلات جديدة أو بحذف بعض السجلات الموجودة.

ويعيد استعلام الأداء عدد السجلات التي تأثرت بالعملية (وليس السجلات نفسها).

| **ملحوظة:** |
|---|
| لن تستطيع تجربة هذه الاستعلامات في نافذة باني الاستعلام الخاصة بالعروض View، لأنها كما ذكرنا تستخدم استعلامات التحديد لعرض جزء من جدول أو أكثر.. وبدلا من هذا، يمكنك ضغط العنصر Views في متصفح الخوادم Server Explorer بزر الفأرة الأيمن وضغط الأمر New Query لفتح نافذة باني استعلام خاصة باستعلام عام.. في هذه الحالة يمكنك تجربة أمثلة استعلامات الأداء بلصقها في جزء الاستعلام SQL PANE في هذه النافذة، وضغط الأمر Execute SQL من القائمة الموضعية. |

## حذف الصفوف باستخدام الأمر DELETE:

تحذف هذه الجملة أيّ عدد تريده من الصفوف تبعا للشرط الذي تحدّده في المقطع WHERE.. والجملة التالية تريك كيف يمكن حذف كل الكتب نفدت:

```
DELETE FROM Books
WHERE Copies_No = 0
```

| **ملحوظة:** |
| --- |
| لبناء أمر الحذف السابق في باني الاستعلام، أضف الجدول Books إلى قسم المخطط، واضغط بزر الفأرة الأيمن في أي موضع خال من قسم المخطط، ومن القائمة الموضعية اضغط Change Type ومن القائمة الفرعية اضغط الأمر Delete.. ستجد أن جملة SQL المكتوبة قد تحولت إلى DELET بدلا من SELECT.<br>استخدم قسم المعايير Criteria Pane لإضافة اسم الحقل Copies_No، وضع في العمود Filter القيمة:<br>= 3 |

والجملة التالية تحذف كل الكتب التي كتبها (عباس العقاد):

```
DELETE FROM Books
WHERE AuthorID = (SELECT ID
                  FROM Authors
                  WHERE Author = 'عباس العقاد')
```

طبعا لاحظت استخدامنا لاستعلام التحديد في جملة الشرط.. هذه الإمكانيّة تمنحك قدرات بلا حدود، لحذف السجلات التي تنطبق عليها المواصفات التي تحدّدها.

وهناك طريقة أخرى لأداء نفس العملية، وذلك باستخدام الفقرة From لاختيار الجداول ووضع شروط الربط بينها باستخدام الفقرة WHERE أو طرق الربط المختلفة Joins.. دعنا نكتب المثال السابق باستخدام الربط الداخلي:

```
DELETE FROM Books
FROM Books INNER JOIN
      Authors ON Books.AuthorID = Authors.ID
WHERE (Authors.Author = 'عباس العقاد')
```

وهو نفس ما يمكنك فعله بالجملة WHERE كالتالي:

```
DELETE FROM Books
FROM   Books, Authors
WHERE AuthorID = Authors.ID AND
Authors.Author = 'عباس العقاد'
```

| ملحوظة: |
| --- |
| لبناء أمر الحذف السابق في باني الاستعلام، اضغط بزر الفأرة الأيمن في أي موضع خال من قسم المخطط، ومن القائمة الموضعية اضغط Change Type ومن القائمة الفرعية اضغط الأمر Delete.<br>أضف الجدول Books أولا (ليتم الحذف منه) ثم أضف الجدول Authors ليقوم باني الاستعلام بتكوين فقرة الربط بينهما.<br>وفي قسم المعايير Criteria Pane اختر اسم الحقل Authors.Author من العمود Column وفي العمود Filter الخاص بهذا الحقل ضع الشرط:<br>'عباس العقاد' = |

## إدراج سجلات جديدة باستخدام الأمر INSERT:

يمكنك إضافة سجلات جديدة إلى الجدول، باستخدام أمر الإدراج INSERT على الصيغة التالية:

**INSERT INTO اسم الجدول (أسماء الحقول) VALUES (قيم الحقول)**

حيث ستضاف القيم إلى الحقول تبعا لترتيب كتابة أسمائها.. ويمكن ألا تكتب كل أسماء الحقول، وفي هذه الحالة ستترك الخانات المناظرة للحقول فارغة.. ويجب أن يكون عدد القيم المرسلة مساويا لعدد أسماء الحقول المكتوبة.. والمثال التالي يضيف سجلا جديدا إلى جدول المؤلفين:

**INSERT INTO Authors (Author, CountryID, About)**
**VALUES ('فاروق جويدة', 21, 'شاعر مصري معاصر')**

لاحظ أننا لم نرسل قيمة إلى الحقل ID لأنه ترقيم تلقائي، ولن يقبل منك أي قيمة.. كما أننا لم نرسل قيمة حقل رقم الهاتف، لأننا صممناه بحيث يقبل القيمة الفارغة NULL، ولا مانع من إرسال قيمة له لو أردت.. لاحظ أيضا أن الرقم ٢١ هو رقم مصر في جدول الدول.

وهناك تسهيل آخر، يتيح لك إدخال قيم كل الحقول دون أن تكتب أسماءها، لكن في هذه الحالة عليك أن تكتب القيم مرتبة تبعا لترتيب الحقول الأصليّ في الجدول، وذلك على الصيغة التالية:

**INSERT INTO اسم الجدول VALUES (قيم الحقول)**

فمثلا: لإضافة حقل جديد إلى جدول الكتب، استخدم الجملة التالية:

**INSERT INTO Books**
**VALUES ('كانت لنا أوطان', 14, 6, 7, '1/8/2000', 3, 2, 1000)**

لاحظ أننا لم نضع قيمة للحقل ID لأنه يُولد تلقائيا، لهذا ستوضع القيم بالترتيب في الحقول التالية لهذا الحقل.. لكن المشكلة أنك لو لصقت الجملة السابقة في باني

الاستعلام، فسيضيف أسماء الحقول تلقائيا إليها، ومن ضمنها الحقل ID، لهذا عليك أن تحذفه وإلا حدث خطأ عند تنفيذ الاستعلام.

لاحظ أيضا ضرورة وضع التاريخ بين العلامتين ' ' عند كتابة جمل SQL، تماما كما نفعل مع النصوص.. كما أنك تستطيع استخدام الكلمة NULL كقيمة، إذا أردت ترك الخانة فارغة، وإذا كانت الخانة تحتوي على بيانات ثنائية (كالصورة image) فأرسل إليها القيمة 0x0 وهي تعادل العدم NULL.

| ملحوظة: |
|---|
| لبناء أمر الإدراج في باني الاستعلام، أضف الجدول الذي ستقوم بالإدراج فيه إلى قسم المخطط، واضغط بزر الفأرة الأيمن في أي موضع خال من قسم المخطط، ومن القائمة الموضعية اضغط Change Type ومن القائمة الفرعية اضغط الأمر Insert Values.. ستجد أن جملة SQL المكتوبة قد تحولت إلى INSERT بدلا من SELECT، وستكون على هذه الصورة:<br>**INSERT INTO اسم الجدول ( )**<br>**VALUES ( )**<br>اختر أسماء الحقول التي تريد الإدراج فيها بوضع علامة الاختيار بجوارها في مخطط الجدول، لتظهر أسماؤها في جملة SQL داخل القوسين الفارغين التاليين لاسم الجدول.. بعد هذا عليك أن تضيف القيم بكتابتها في العمود New Value في قسم المعايير Criteria Pane، حيث ستكتب القيمة الجديدة في الخانة المجاورة لاسم كل حقل.<br>ولو أردت العودة إلى استخدام الأمر SELECT، فاضغط بزر الفأرة الأيمن في أي موضع خال من قسم المخطط، ومن القائمة الموضعية اضغط Change Type ومن القائمة الفرعية اضغط الأمر Select. |

ويتيح لك الأمر INSERT نسخ مجموعة من السجلات من جدول إلى آخر، بشرط أن يكون لحقولهما نفس نوع البيانات، وذلك باستبدال الفقرة VALUES بالجملة SELECT.

افترض أن عندنا جدولا اسمه TempBooks مماثلا في تركيبه لجدول الكتب، ونريد أن نضع فيه بعض الكتب مؤقتا لأيّ سبب.. في هذه الحالة يمكن أن ننسخ فيه كتب (توفيق الحكيم) بالجملة التالية:

```
INSERT INTO TempBooks
SELECT Books.*
FROM Authors, Books
WHERE Author = 'توفيق الحكيم' AND AuthorID = Authors.ID
```

| ملحوظة: |
| --- |
| لبناء الأمر الأخير في باني الاستعلام، أضف الجدولين Authors و Books إلى قسم المخطط، وحدد كل الحقول من جدول الكتب، وحدد الحقل Author من جدول المؤلفين، وفي قسم المعايير Criteria Pane أزل علامة الاختيار من العمود Output في صف الحقل Author لكي لا يظهر في النتائج، وفي الخانة Filter الخاصة بهذا الحقل أضف الشرط: ( 'توفيق الحكيم' = )<br>الآن لديك جملة التحديد SELECT التي تحصل على النتائج التي تريد إدراجها في الجدول TempBooks.. كل ما عليك فعله الآن، هو الضغط بزر الفأرة الأيمن في أي موضع خال من قسم المخطط، ومن القائمة الموضعية اضغط Change Type ومن القائمة الفرعية اضغط الأمر Insert Results.. ستظهر لك نافذة تتيح لك اختيار الجدول الذي تريد إدراج النتائج فيه.. اختر الجدول TempBooks وأغلق النافذة.. الآن ستجد جملة الإدراج كاملة في قسم الاستعلام SQL Pane! |

## تحرير السجلات الموجودة باستخدام الأمر UPDATE:

لتغيير قيم بعض ــ أو كل ــ سجلات أحد الجداول، استخدم الجملة UPDATE، التي لها الصيغة التالية:

**UPDATE اسم الجدول**
**SET الحقل١ = القيمة١, الحقل٢ = القيمة٢, ........**
**WHERE شرط**

مثال: الجملة التالية تغير اسم المؤلف (نبيل فاروق) إلى (نبيل فاروق رمضان):

**UPDATE Authors**
**SET Author = 'نبيل فاروق رمضان'**
**WHERE Author = 'نبيل فاروق'**

| ملحوظة: |
| --- |
| لبناء الأمر الأخير في باني الاستعلام، أضف الجدول Authors إلى قسم المخطط، وحدد الحقل Author.<br>اضغط بزر الفأرة الأيمن في أي موضع خال من قسم المخطط، ومن القائمة الموضعية اضغط Change Type ومن القائمة الفرعية اضغط الأمر Update، وفي قسم المعايير Criteria Pane أضف الشرط التالي في الخانة Filter الخاصة بالحقل Author: ('نبيل فاروق' =)<br>وفي العمود New Value السابق للعمود Filter اكتب: 'نبيل فاروق رمضان'<br>الآن ستجد الأمر UPDATE مكتوبا في قسم SQL كاملا. |

## التحرير المتتابع للسجلات، باستخدام الأمر UPDATE .WRITE:

عند التعامل مع أعمدة تحتوي على بيانات من النوع text أو ntext أو image في سيكيويل سيرفر ٢٠٠٠، كانت تواجهنا مشكلة خطيرة، وهي كيفية إرسال كم ضخم من البيانات من العميل إلى الخادم لحفظه في هذه الأعمدة.. افرض أنك تريد وضع صورة حجمها ١٠ ميجا في خانة من النوع image بالاستعلام التالي:

**UPDATE Publishers**
**SET Logo = @Logo**
**WHERE ID = 1**

حيث Logo@ هو معامل من النوع image يحمل بيانات الصورة.. لو حاولت تنفيذ هذا الاستعلام، فسيستغرق إرسال الصورة إلى الخادم وقتا ملموسا (تبعا لسرعة الاتصال، وحجم الضغط على الخادم في تلك اللحظة)، وقد يتعطل برنامجك عن الاستجابة، وقد ينفد وقت الانتظار Timeout قبل إتمام العملية.

ويمكنك أن تجرب الاستعلام السابق بضغط الزر Add Logo في المشروع WriteLargeDate.. جرب اختيار صورة حجمها كبير لترى تأثير هذا.

وللأسف، لا يفيدك استخدام استعلام كالتالي:

**UPDATE Publishers**
**SET Logo = Logo + @Logo**
**WHERE ID = 1**

هذا الاستعلام يحاول إضافة جزء من بيانات الصورة موضوع في المعامل Logo@ إلى البيانات الموجودة فعليا في عمود الصورة Logo.. لكن للأسف، سيؤدي هذا إلى حدوث خطأ، لأن سيكويل سيرفر لا يعرف كيف يجمع بيانات من النوع image!

ولحل هذه المشكلة، قدمت سيكويل سيرفر ٢٠٠٥ الأنواع القصوى الجديدة varchar(max), nvarchar(max), or varbinary(max) وهي مخصصة للتعامل مع الكائنات الثنائية الضخمة (BLOBs) Binary Large Objects، فهي تستطيع استقبال بيانات يصل حجمها إلى حوالي ٢ جيجا بايت، كما يمكنها التعامل مع البيانات الثنائية والحروف بطريقة تتابعية Sequential، أي أنك تستطيع الكتابة فيها في أي موضع، أو القراءة منها من أي موضع.

لهذا إذا كنت تنوي التعامل مع بيانات ضخمة BLOBs، فالأفضل أن تستخدم:

- النوع varchar(max) بدلا من النوع Text.
- والنوع nvarchar(max) بدلا من النوع ntext.
- والنوع varbinary(max) بدلا من النوع image.

وللكتابة التتابعية في هذه الأنواع الجديدة، قدمت T-SQL الصيغة الجديدة التالية لأمر التحديث:

UPDATE اسم_الجدول
SET اسم_العمود .WRITE (@Value, @Offset, @Length )
Where شرط

فالدالة الداخلية Write تكتب البيانات المرسلة إلى المعامل Value@ في الخانة المحددة في العمود، بدءا من الموضع Offset@، وبحيث يكون طول البيانات المكتوبة Length@.. لاحظ ما يلي:

- هذه الصيغة ستسبب خطأ إذا حاولت التعامل مع أي نوع بيانات غير الأنواع القصوى التي تنتهي بـ (MAX).
- هذه الصيغة ستسبب خطأ إذا حاولت إضافة بيانات في خانة قيمتها Null.. لهذا أمامك حلان:

١- فإما أن تضيف الجزء الأول من البيانات باستخدام الصيغة العادية للأمر Update، ثم تضيف باقي أجزاء البيانات باستخدام الصيغة Update .Write.

٢- وإما أن تمنع استخدام القيمة Null في خانات العمود، وفي هذه الحالة عليك استخدام قيمة افتراضية لوضعها في الخانات الفارغة، وذلك بوضع القيمة صفر في الخاصية Default Value Or Binding في خصائص العمود.. ولكي لا تؤثر هذه القيمة الافتراضية على القيمة التي ستضعها في خانات العمود، يجب أن تكون للمعامل Offset@ القيمة صفر عند كتابة أول جزء من البيانات، لتوضع في بداية الخانة بدلا من أية بيانات موجودة.

- إذا أرسلت إلى المعامل Value@ القيمة Null، فسيتم تجاهل المعامل Length@، وسيتم حذف القيمة الموجودة في الخانة الموضع المحدد في المعامل Offset@.
- إذا أرسلت إلى المعامل Offset@ القيمة Null، فسيتم تجاهل المعامل Length@، وستضاف قيمة المعامل Value@ إلى نهاية البيانات الموجودة حاليا في الخانة.
- إذا أرسلت إلى المعامل Offset@ قيمة أكبر من طول البيانات الموجودة في الخانة، فسيحدث خطأ.. ولكي تكتب بعد نهاية البيانات الموجودة، يجب أن ترسل إلى المعامل Offset@ قيمة تساوي طول البيانات الموجودة في الخانة، أو ترسل إليه القيمة Null كما أوضحنا في الملحوظة السابقة.
- إذا أرسلت إلى المعامل Length@ القيمة Null، فسيتم حذف جميع البيانات التالية للموضع المحدد في المعامل Offset@.

وستجد مثالا لاستخدام هذه الصيغة لكتابة بيانات صورة أول ناشر في العمود Logo2 في الجدول Publishers بطريقة تتابعية، وذلك في الزر Update .Write في المشروع WriteLargeData، وفيه تستخدم الاستعلام:

```
UPDATE Publishers
SET Logo2 .WRITE (@Logo, @Offset , @Length)
WHERE ID = 1
```

لاحظ أن العمودLogo2 من النوع varbinary(MAX)، وأن قيمته الافتراضية هي ٠.. ولكتابة بيانات الصورة في الخانة الأولى من هذا العمود تتابعيا، سنرسل أول ١٠٠ وحدة ثنائية لحفظها في الخانة بدءا من الموضع رقم صفر، ثم نرسل ١٠٠ وحدة تالية Byte لحفظها في الخانة بدءا من الموضع رقم ١٠٠، ونستمر في فعل هذا إلى أن ننتهي من كتابة بيانات الصورة.. هذا معناه أننا سنستخدم الاستعلام Write. Update عدة مرات (وذلك من خلال حلقة تكرار Loop)، لكن مع تغيير المعاملات المرسلة إلى الدالة الداخلية Write. في كل مرة.. طبعا إرسال ١٠٠ وحدة ثنائية في كل مرة، أفضل بكثير من إرسال ١٠ ميجا أو أكثر دفعة واحدة.. لكن هذا قد يصير عبئا خطيرا على البرنامج إذا كانت الصورة ضخمة جدا، بسبب زيادة عدد مرات إرسال البيانات من العميل إلى الخادم.. لهذا عليك اختيار حجم مناسب لأجزاء البيانات التي ترسلها، بحيث لا يكون كبيرا جدا فيدمر الذاكرة ويبطئ عملية الإرسال، ولا يكون صغيرا جدا فيؤدي إلى إضاعة وقت كبير من الخادم بسبب كثرة عدد الأوامر المرسلة إليه من عميل واحد.. ربما يكون الأنسب مثلا أن تستخدم ١٠٢٤ وحدة ثنائية (١ كيلو بايت) لكل جزء.. لكن هذا سيجعلك ترسل الصورة التي حجمها ١ ميجا فقط على حوالي ١٠٠٠ مرة.. ما زال هذا يبدو كثيرا.. أليس كذلك؟.. لو شئت رأيي، فإن إرسال ١٠٠ كيلو في كل مرة سيكون مناسبا، فهذا سيرسل الصورة التي حجمها ١ ميجا باستخدام ١٠ أوامر فقط وهذا ليس كثيرا، كما أن ١٠٠ كيلو ليس بالحجم المقلق الذي يستغرق وقتا ملموسا عند إرساله إلى الخادم.. لكن عليك في هذه الحالة أن تجعل للخاصية Timeout الخاصة بالتصال الذي تستخدمه قيمة أكبر قليلا، ولتكن ١٢٠ ثانية على سبيل الاحتياط.. وسنعرف كيف نفعل هذا لاحقا عند دراسة كائن الاتصال وكائن الأمر.

| **ملحوظة:** |
|---|
| الإصدارات القديمة من T-SQL، كانت تستخدم الأوامر UPDATETEXT و READTEXT و WRITETEXT للتعامل مع البيانات الضخمة الموجودة في الأعمدة من النوع image أو text أو ntext.. لكن لا ينصح باستخدام هذه الأوامر الآن لأنها ستزال من لغة الاستعلام، وبدلا من هذا عليك استخدام الأنواع القصوى، والصيغة Write. Update. |

## لغة تعريف البيانات Data Definition Language (DDL):

تختصّ أوامر هذه اللغة بإنشاء أو حذف كائنات قاعدة البيانات، مثل الجداول وما تحتويه من أعمدة وفهارس، ومثل العلاقات Relations والقيود Constrains.
ولن نتوسّع في شرح هذه الأوامر في هذا الكتاب، وسنتركها إلى الكتاب الذي نشرح فيه T-SQL بإذن الله، ويمكنك أن تبحث في ملفات استعلام اللغة عن كيفيّة استخدام الأوامر التالية، وهي تستخدم لإنشاء عناصر قاعدة البيانات:

**CREATE DATABASE**
**CREATE TABLE**
**CREATE PROCEDURE**
**CREATE FUNCTION**
**CREATE VIEW**
**CREATE INDEX**

ولتعديل عناصر قاعدة البيانات، تستخدم الكلمة ALTER بدلا من الكلمة CREATE.. ولحذف أحد العناصر، تستخدم الكلمة DROP بدلا من الكلمة CREATE.

ويمكنك الاسترشاد بهذه الأمثلة (وهي خاصة بقواعد بيانات Access):
الجملة التالية تنشئ جدولا للمؤلفين، به عمودان يستوعبان نصوصا لا يزيد طولها عن ٣٠ حرفا، مع ملاحظة أن العمود الثاني مفهرس (بإنشاء فهرس اسمه X) مع عدم السماح بتكرار القيم:

```
Create Table Authors
      ([Author] Text  (30), [City] Text (20) Constraint X
      Unique)
```

أمّا الجدول التالي، فله عمود رقميّ، والآخر ترقيم تلقائيّ:

```
Create Table Numbers
      ([No] Integer,[ID] Counter)
```

ولإنشاء فهرس اسمه X على العمود ID في الجدول السابق:

```
Create Unique Index X On Numbers (No)
```

ولإضافة عمود اسمه Y للجدول السابق:

```
ALTER TABLE  Numbers
      ADD COLUMN Y Long
```

ولحذفه مرّة أخرى:

```
ALTER TABLE  Numbers DROP COLUMN Y
```

## الإجراءات المخزّنة Stored Procedures:

الإجراءات المخزّنة هي برامج قصيرة مكتوبة بلغة T-SQL محفوظة في قاعدة البيانات، ويتم تنفيذها على الخادم Server، لأداء وظيفة معيّنة على قاعدة البيانات.

ويمتاز الإجراء المخزن بالمميزات التالية:

١- يقوم SQL Server بترجمة Compile الإجراءات المخزّنة مرّة واحدة عند إنشائها، وبالتالي فهي تعمل أسرع من جمل SQL العاديّة.

٢- اختصار الكود، بسبب قدرتك على استدعاء نفس الإجراء المخزن من برامج مختلفة أو من أكثر من موضع من نفس البرنامج.

٣- سهولة التعديل في الإجراء المخزن بدون تغيير أكواد البرامج التي تستخدمه، وذلك لأنه موجود في قاعدة البيانات.

٤- الإجراءات المخزنة أكثر مناعة ضد دس الاستعلامات SQL Injection كما سنرى فيما بعد.

وتسمّى اللغة التي تكتب بها الإجراءات المخزّنة T-SQL اختصارا لـ (Transact-SQL)، وهي نسخة مطوّرة من SQL خاصة بسيكيول سيرفر كما ذكرنا سابقا.

ويبدأ الإجراء المخزن بجملة التعريف التالية:

```
CREATE PROCEDURE dbo.SP1
```

حيث إن SP1 هو اسم الإجراء، ويمكنك تغييره إلى أيّ اسم يناسبك.

لاحظ أن الجملة CREATE PROCEDURE تستخدم عند إنشاء الإجراء المخزن للمرة الأولى، لكن بمجرد حفظه تتحول إلى ALTER PROCEDURE، لأنك ستعدل فيه بعد ذلك ولن تعيد إنشاءه من جديد.. ولو حاولت استخدام CREATE PROCEDURE مع نفس اسم الإجراء المخزن مرة أخرى فستحصل على خطأ يخبرك أن هذا الإجراء موجود من قبل، لهذا لو كنت تقصد إعادة حفظ نفس الإجراء بعد التعديل، فتأكد من أنه يبدأ بالفقرة ALTER PROCEDURE.

بعد هذا يأتي تعريف معاملات الإجراء.. وتبدأ المتغيّرات في لغة T-SQL بالرمز @.. والجملة التالية تعرّف معاملا نصّيا طوله ثلاثة أحرف:

```
@Str1 nvarchar(3)
```

كما ترى: تكتب اسم المعمل، يليه نوعه.

وعند وجود أكثر من معامل، توضع العلامة "," للفصل بينها.

ويمكن أن تضع للمعامل قيمة افتراضيّة، بحيث لو لم يُرسل هذا المعامل، يقوم الإجراء باستخدام هذه القيمة:

**@Str2 nvarchar(7) = 'افتراضي'**

وتوضع معاملات الإجراء بينَ قوسين ()، ولا يشترط أن تكتب في سطر واحد.

ويبدأ الإجراء المخزّن بالكلمة AS، وينتهي بالكلمة RETURN، وبينهما الكود:

```
AS
...........
RETURN
```

ويمكنك أن تعرّف المتغيّرات داخل الإجراء، باستخدام الجملة DECLARE:

```
DECLARE @Str3 nvarchar(7)
```

ويمكنك كتابة جمل الشرط كالتالي:

```
IF @Str3 = ''
BEGIN
   SELECT @Str3 = 'ABC'
END
```

وإذا كان مقطع الشرط جملة واحدة، فلسنا بحاجة إلى BEGIN و END:

```
IF @Str3 = ''
  SELECT @Str3 = 'ABC'
```

لاحظ استخدامنا للتعبير SELECT لتغيير قيمة المتغيّر Str3@.. يمكنك كذلك استخدام التعبير SET لتغيير قيمة المتغيّر، وهو ليس مربكا مثل SELECT:

```
SET @Str3 = 'ABC'
```

ويمكن أن تضع في المتغيّر ناتج جملة SELECT ــ طبعا إذا كانت تعيد قيمة واحدة، كأن تكون بها دالة من دوال التجميع Aggregate Functions أو يكون الجدول الناتج عبارة عن خانة واحدة ــ كالتالي:

```
SET @Str3 = (SELECT Book FROM Books WHERE ID = 3)
```

والآن، ما رأيك أن نكتب إجراء مخزّنا نرسل إليه اسم المؤلف، فيعيد إلينا الكتب التي ألفها؟

في متصفح خوادم الإنترنت Server Explorer، أسدل العناصر التي تنتمي لاتصال قاعدة البيانات Books.mdf التي أنشأناها بـ SQL Sever.. من هذه العناصر اضغط بزرّ الفأرة الأيمن على "الإجراءات المخزّنة" Stored Procedures، ومن القائمة الموضعيّة اضغط الأمر New Stored Procedure.. ستظهر لك نافذة تحرير الإجراء المخزّن.. اكتب بها هذا الإجراء:

```
CREATE PROCEDURE dbo.GetAuthorBooks
	(
		/*لدينا معامل واحد فقط نستقبل فيه اسم المؤلف*/
		/* وقد جعلناه ۳۰ حرفا ليتلاءم مع طول حقل المؤلفين */
		@Author char(30)
	)
AS
IF @Author <>''	/* تأكّد أن اسم المؤلف غير فارغ */
BEGIN
	/*هذه جملة استعلام عاديّة تماما*/
	SELECT Book
	FROM Books, Authors
	WHERE Author = @Author
		AND AuthorID = Authors.ID
END
RETURN /*وجود هذه الكلمة في أيّ موضع يؤدّي إلى إنهاء الإجراء*/
```

أعتقد أن الأمر في غاية البساطة.

ولا حاجة بي لألفت نظرك إلى أن التعليق في لغة T-SQL يوضع بين العلامات /* */.. هذه الرموز تتيح لك كتابة التعليق على أكثر من سطر.. فإذا أردت أن تكتب سطرا واحدا كتعليق، فيمكنك أن تضع في بدايته العلامتين --.. مثال:

```
/* هذا تعليق
على سطرين*/
-- بينما هذا تعليق في سطر واحد
```

ولا بدّ أنك لاحظت ظهور الجملة SELECT في نافذة الإجراء السابق في مستطيل.. اضغط بزرّ الفأرة الأيمن داخل هذا المستطيل، ومن القائمة الموضعيّة اختر الأمر "تصميم مقطع الاستعلام" Design SQL Block.. ستظهر لك نافذة باني الاستعلام Query Builder، حيث يمكنك استغلاله لتطوير الجملة واختبارها.

كما يمكنك استخدام باني الاستعلام لتصميم جملة SQL مباشرةً منذ البداية.. اضغط بزرّ الفأرة الأيمن في الموضع الذي تريد كتابة الجملة فيه في الإجراء المخزّن، ومن القائمة الموضعيّة اضغط الأمر Insert SQL.. ستظهر لك نافذة باني الاستعلام.. صمّم جملة SQL التي تريد، وقم بحفظ العمل.. الآن لو عدت إلى نافذة الإجراء المخزّن، فستجد أن جملة SQL قد أضيفت للموضع الذي ضغطت فيه الفأرة.

نريد الآن اختبار الإجراء الذي كتبناه.. اضغط بزرّ الفأرة الأيمن في أيّ موضع من نافذة الإجراء، ومن القائمة الموضعيّة اضغط الأمر Run Stored Procedure.. ستظهر لك نافذة تسألك عن قيمة معامل الإجراء.. اكتب اسم (توفيق الحكيم) واضغط OK.. سيتمّ تنفيذ الإجراء، حيث ستظهر أسماء كتب (توفيق الحكيم) في نافذة المخرجات Output Window.

## إنشاء الإجراءات المخزنة في قواعد بيانات Access:

على عكس ما يظنه الكثيرون، يمكن استخدام الإجراءات المخزنة مع قواعد بيانات آكسيس.. لكن المشكلة أنك لن تجد طريقة لإنشائها في آكسيس نفسه، كما أنك لا تستطيع إنشائها في متصفح الخوادم Server Explorer في دوت نت، رغم أنك ستجد عنصرا اسمه Stored Procedures ضمن عناصر قاعدة بيانات الكتب Books.mdb!

فما هو حل هذا اللغز العجيب يا ترى؟

الحل هو تنفيذ الجملة التي تنشئ الإجراء المخزن من خلال كود فيجيوال بيزيك في أحد برامجك لمرة واحدة، وبهذا يتم إنشاء الإجراء المخزن في قاعدة بيانات آكسيس، ويظهر في متصفح الخوادم تحت العنصر Stored Procedures.. طبعا لا نستطيع فعل هذا الآن، لهذا سنؤجله إلى فصل لاحق.. لكن علينا هنا أن نوضح بعض الاختلافات الجوهرية بين الإجراء المخزن الخاص بآكسيس والإجراء المخزن الخاص بسيكويل سيرفر:

١- يستخدم آكسيس التعبير CREATE PROC بدلا من التعبير CREATE PROCEDURE.

٢- الإجراء المخزن في آكسيس يتكون من جملة SQL واحدة فقط، لأن آكسيس لا يقبل تنفيذ أكثر من جملة مفصولة بالعلامة ;.

٣- لا يمكن تعريف متغيرات في الإجراء المخزن في آكسيس ولا يمكن استخدام جمل الشرط.. السبب ببساطة أن آكسيس لا يتعامل مع T-SQL.

٤- لا توضع العلامة @ قبل أسماء المعاملات الخاصة بالإجراء المخزن في آكسيس.. وطبعا أنواع هذه المعاملات هي الأنواع الخاصة بآكسيس.

٥- لا ينتهي الإجراء المخزن في آكسيس بالكلمة RETURN.

باختصار: الإجراء المخزن في آكسيس، هو مجرد استعلام سيكويل مخزن!

هكذا مثلا سيكون الإجراء المخزن الذي يحذف من جدول المؤلفين، المؤلف الذي ترسل اسمه إليه:

```
CREATE PROC DeleteAuthor(
                    AuthorName VARCHAR(20))
AS
DELETE FROM Authors
        WHERE (Authors.Author = AuthorName);
```

ولكي تضيف هذا الإجراء على قاعدة بيانات الكتب، افتح المشروع AccessStoredProcedure، وشغل البرنامج، واضغط الزر "DeleteAuthor".. كما يمكنك إنشاء إجراء مخزن للحصول على كتب أحد المؤلفين، بضغط الزر "GetAuhorBooks".

لاحظ أن هذا البرنامج يفترض وجود الملف Books.mdb على المسار \:C.. وسنفهم الكود الذي يستخدمه هذا البرنامج بالتفصيل لاحقا.

بعد أن تنفذ هذا البرنامج، أضف اتصالا بقاعدة بيانات الكتب الموجودة على المسار \:C في متصفح الخوادم، وأسدل عناصرها.. ستجد أن الإجراءين DeleteAuthor و GetAuhorBooks قد ظهرا تحت العنصر Stored Procedures.. لكن للأسف، لا يمنحك متصفح الخوادم أية طريقة لعرض كود هذين الإجراءين أو تعديلهما أو تشغيلهما.

ويريك الزر "الكتب١" في المشروع Factories كيف يمكن استدعاء الإجراء GetAuhorBooks من برنامجك، وسنفهم كيف يفعل هذا لاحقا.

## دوال SQL التي يعرفها المستخدم User Defined SQL Functions:

يمكنك تعريف دوال خاصة، تستخدمها داخل استعلامات SQL، بنفس الطريقة التي استخدمنا بها الدالة RIGHT.. هذه الدوال تحفظ في قاعدة البيانات تماما كالإجراءات المخزنة، وتظهر في متصفح الخوادم تحت العنصر Functions.

ويمكنك إنشاء دالة جديدة بإسدال عناصر قاعدة البيانات وضغط العنصر Functions بزر الفأرة الأيمن، وضغط القائمة الفرعية New، حيث ستظهر لك ثلاثة أنواع من الدوال التي تستطيع إنشاءها.. لكن نظرا لأن هذا الموضوع يحتاج إلى تفاصيل أعمق في T-SQL فسنؤجله إلى الوقت المناسب بإذن الله.

## أنواع الجداول التي يعرفها المستخدم User-Defined Table Types:

قدم سكيويل سيرفر ٢٠٠٨ نوعا جديدا من المعاملات، قادرا على استقبال جدول كامل يحتوي على مجموعة صفوف، بدلا من المعاملات العادية التي تستقبل قيمة خانة منفردة في أحد الصفوف.. هذا يغنيك عن استخدام عدد كبير من المعاملات عند تعريف الإجراء المخزن، كما يغنيك عن استدعاء نفس الإجراء المخزن أكثر من مرة.. فمثلا: لو أردت كتابة إجراء مخزن يقوم بتحديث قيم ٥ صفوف بها عشرة أعمدة في أحد الجداول، فإن استخدام المعاملات التقليدية سيجبرك على تعريف ١٠ معاملات للإجراء المخزن (واحد لكل عمود)، واستخدام واستدعاء الإجراء المخزن ٥ مرات لإرسال القيم إلى هذه الصفوف.. بينما إذا استخدمت معاملا جدوليا Table-Valued Parameter، فستعرف معاملا واحدا فقط للإجراء المخزن، وستستدعيه مرة واحدة فقط لتحديث الصفوف الخمسة.

يبدو هذا مريحا للغاية، حيث سيوفر عليك كتابة الكثير من الكود في الإجراء المخزن وفي برنامجك أيضا، وسيجعل نقل كميات كبيرة من البيانات أمرا في غاية البساطة.

ولاستخدام هذا النوع من المعاملات، يجب أن تقوم بتعريفه أولا في قاعدة البيانات.. ويشبه تصميم النوع تصميم الجدول، لكن للأسف، لا توجد طريقة مرئية لتصميم النوع، لهذا عليك كتابة كود T-SQL لفعل هذا، وتنفيذه من نافذة تنفيذ الاستعلامات في مدير سيكويل.. ويستخدم الأمر CREATE TYPE لإنشاء نوع جديد في T-SQL، وذلك على الصيغة:

**CREATE TYPE اسم_النوع AS TABLE**
**(تعريف_الأعمدة)**

دعنا ننشئ نوعا اسمه AuthorType نستطيع من خلاله تحديث صفوف جدول المؤلفين.. لفعل هذا في مدير سيكويل، اضغط اسم قاعدة البيانات Books.mdf بزر الفأرة الايمن، ومن القائمة الموضعية اضغط الأمر New Query، وفي نافذة الاستعلام اكتب الكود التالي:

```
CREATE TYPE AuthorType AS  TABLE
  ( ID int,
    Author nvarchar(50),
    CountryID smallint,
    Phone varchar(20),
    About nvarchar(MAX) )
```

من القائمة الموضعية اضغط Execute.. الآن سيظهر النوع AuthorType تحت العنصر Programmability\Types\User defined Table Types

في متصفح الكائنات في مدير سيكويل سيرفر، حيث يمكنك تغيير اسمه لو أردت، كما يمكنك حذفه في أي وقت.
الآن يمكنك تعريف إجراء مخزن جديد يستخدم معاملا من هذا النوع.. لاحظ أنك لا تستطيع استخدام هذا المعامل للإخراج، كما أنك مجبر على تعريفه للقراءة فقط باستخدام الكلمة READONLY كالتالي:

```
CREATE PROCEDURE UpdateAuthors
  (@Rows AuthorType READONLY)
```

لكن كيف نتعامل مع المعامل @Rows في كود الإجراء المخزن؟
الأمر بسيط للغاية، فهذا المعامل هو جدول، ويمكنك التعامل معه في جمل SQL كأي جدول عادي.. مثلا: يمكننا تحديث الجدول Authors من الجدول @Rows باستخدام الأمر UPDATE والربط الداخلي INNER JOIN كالتالي:

```
CREATE PROCEDURE UpdateAuthors
  (@Rows AuthorType READONLY)
AS
UPDATE Authors
  SET Authors.Author = R.Author,
      Authors.CountryID = R.CountryID,
      Authors.Phone = R.Phone,
      Authors.About = R.About
  FROM Authors INNER JOIN @Rows AS R
  ON Authors.ID = R.ID;
RETURN
```

لاحظ ضرورة تعريف اسم مستعار للجدول @Rows في المقطع FROM لتسهيل كتابة الاستعلام.. لقد منحناه هنا الاسم المستعار R.
بهذا يمكنك استخدام هذا الإجراء المخزن لتحديث قيم سجلات مؤلفين موجودين بالفعل في قاعدة البيانات، بناء على تماثل الرقم ID في السجل المرسل والسجل الأصلي.
كما يمكنك تعريف إجراء لإدراج سجلات الدول المرسل كمعامل، في جدول المؤلفين، وذلك باستخدام الأمر INSERT.. هذا هو:

```
CREATE PROCEDURE InsertAuthors
   (@Rows AuthorType READONLY)
AS
INSERT INTO  Authors
   (Author, CountryID, Phone, About)
SELECT R.Author, R.CountryID, R.Phone, R.About
FROM @Rows AS R
RETURN
```

لاحظ أنك لا تستطيع اختصار جملة الإدارج السابقة كالتالي:

```
INSERT INTO  Authors
SELECT R.*
FROM @Rows AS R
```

السبب في هذا هو أنك لا تستطيع نسخ قيمة الحقل ID من سجلات المعامل إلى سجلات الجدول الأصلي، لأن الحقل Author.ID حقل يولد تلقائيا، بينما نحن لم نعرف الحقل AuthorType.ID على أنه يولد تلقائيا.

لاحظ أن تعريف المعامل الجدول بالكلمة ReadOnly يمنعك عن إجراء أية تعديلات على قيم سجلاته، كما يمنعك من حذف أي من هذه السجلات أو الإضافة إليها.. لهذا إذا كنت مضطرا إلى فعل هذا، فعليك أن تعرف متغيرا من نفس نوع الجدول، وتدرج فيه سجلات المعامل باستخدام الأمر INSERT.. مثال:

```
DECLARE @TEMP AuthorType;
INSERT INTO  @TEMP
SELECT R.*
FROM  @Rows AS R;
```

لاحظ أننا استخدمنا صيغة الإدراج المختصرة (لم نكتب أسماء الأعمدة) لأن الحقل ID لا يولد تلقائيا هنا.

بعد هذا يمكنك أن تضيف إلى الجدول TEMP@ سجلات جديدة أو تعدل قيم السجلات الموجودة، ومن ثم تستخدمه في الكود كما تريد.

والمشروع TableValuedParameters يريك كيف تستخدم كائن الأمر والمعاملات لتنفيذ هذا الإجراء المخزن لإدراج عدد من المؤلفين في جدول المؤلفين.. وسنتعرف على كائن الأمر Command ومعاملاته في فصل لاحق.

# -٥-

# تقنية ADO.NET

## الخادم Server والعميل Client:

الخادم Server هو حاسوب توجد عليه قاعدة البيانات، ويعمل عليه SQL Server، لهذا فهو يقوم بقراءة البيانات المطلوبة وإرسالها إلى المستخدم أو استقبال البيانات الواردة من المستخدم وحفظها في قاعدة البيانات.

بينما العميل Client هو أيّ جهاز حاسوب آخر يوجد عليه برنامج قواعد البيانات الذي كتبته أنت، ويقوم بالاتصال بخادم سيكيول لطلب البيانات أو حفظها عليه.

ويمكن أن يكون هناك آلاف العملاء Clients، كلّ منهم يحاول الاتصال بقاعدة البيانات على الخادم في نفس الوقت، وطلب البيانات منها لمعالجتها على أجهزتهم، ثمّ إرسال أيّ تعديلات تمّ إجراؤها عليها إلى الخادم مرّة أخرى، ليتمّ حفظها في قاعدة البيانات.

ويقدم لنا نموذج الخادم والعميل الميزات التالية:

- وجود قاعدة البيانات على الخادم يوفر لمستخدميها مساحة التخزين (بدلا من وضعها على أجهزة كل المستخدمين)، خاصّة حينما تكون قاعدة البيانات عملاقة.
- وجود قاعدة البيانات على الخادم يترك للجهة المسئولة عنها مهمّة تحديثها باستمرار، وهو أفضل من اضطرار كلّ مستخدم إلى شراء نسخة حديثة من قاعدة البيانات كلّ فترة.
- وجود قاعدة البيانات على الخادم يضمن مشاركتها بين مئات المستخدمين، مما يضمن مساهمتهم في إضافة البيانات وقدرة كل منهم على رؤية التعديلات التي أجراها الآخر، بينما لو كانت قاعدة البيانات على جهاز كل منهم بمفرده، فلن يمكنهم العمل الجماعي عليها، وهذا لا يناسب نشاط الشركات التجارية والمؤسسات المالية.
- إجراء العمليّات على البيانات على جهاز العميل بعد الحصول عليها من الخادم، يكون أسرع بكثير من تنفيذ البرنامج على الخادم ثمّ إرسال النّاتج إلى العميل، وذلك لأنّ هناك عددا ضخما من المستخدمين الذين يجرون آلاف العمليّات في نفس اللحظة.

وينظم خادم سيكويل عمليات الاتصال مع العملاء، حيث يخصص لكل اتصال عملية فرعية Thread للقراءة أو الكتابة في قاعدة البيانات.. ويتوقف عدد الاتصالات المتاحة في نفس اللحظة على حجم الذاكرة المؤقتة RAM الموجودة على الجهاز الخادم وقوة المشغل الدقيق الخاص به.. وفي حالة ازدياد الضغط على الخادم يقوم بتأجيل الاستجابة لبعض العملاء إلى حين الانتهاء من خدمة العملاء السابقين، مما قد يؤدي إلى بطء برنامجك وتضايق مستخدميه بسبب تعطله عن الاستجابة لفترات طويلة.. لهذا تقع على برنامجك مسئوليّة ضمان كفاءة عمليات الاتصال بقاعدة البيانات، بمراعاة ما يلي:

- ألا يطلب برنامجك بيانات لا ضرورة لها.. فإن كنت تحتاج مثلا إلى حقل أو حقلين من الجدول، فما الداعي لأن تقرأ كل الحقول؟
- التأكد من كتابة أقصر وأكفأ استعلامات SQL ليكون تنفيذها أسرع فلا ترهق الخادم.
- الاحتفاظ ببعض البيانات المجهزة Cashed على جهاز العميل بدلا من إعادة طلبها أكثر من مرة في فترات زمنية صغيرة، وذلك إذا كنت تضمن عدم تغير هذه البيانات بسرعة كبيرة.. وإذا كانت هذه البيانات ستظل ثابتة لجميع العملاء لفترة طويلة، فيمكن تجهيزها على الخادم وإرسالها إليهم مباشرة كلما طلبوها بدون إعادة تنفيذ الاستعلام، ولا يتم تحديث البيانات المجهزة إلا إذا حدث تغيير فيها في قاعدة البيانات.

مثل هذا التنظيم يضمن تخفيف عبء هائل من على خادم سيكويل وتقليل جمل SQL التي ينفذها، وبالتالي يوفر قدرة المشغل الدقيق Processor والذاكرة RAM الخاصة بالحاسوب الذي يعمل عليه خادم سيكويل، ليستطيع تنفيذ عمليات أخرى.

كما أن هناك بعض التحدّيات التي تواجه المبرمج وهو يكتب برنامجا يتعامل مع الخادم، مثل التعارض الذي يمكن أن ينتج عندما يحذف أحد المستخدمين بعض السجلات، بينما مستخدم آخر يحدّث قيمها!.. أو عندما يحاول أكثر من مستخدم تحديث نفس السجلات بطرق مختلفة في نفس الوقت.. وسنرى كيف نواجه مثل هذا الأمر لاحقا.

## تقنية ADO.NET:

الأحرف ADO هي اختصار المصطلح "كائن البيانات الفعال" ActiveX Data Object، وهي تقنية برمجية ظهرت في فيجيوال ستديو ٦، تقدم جميع الفئات Classes اللازمة للاتصال بقاعدة البيانات وطلب البيانات منها وحفظها فيها.. وتفترض هذه التقنية أنّ العميل سيظل على اتصال بقاعدة البيانات طوال مدّة تعامله معها عبر الشبكة، حيث يحصل على البيانات من أيّ جدول يريده ويحدثها عبر نفس الاتصال.

لكن هذه الطريقة كانت عقيما، ففتح الاتصال طوال الوقت مع قاعدة البيانات عبر الشبكة غير عمليّ لأنه يعطل مستخدمين آخرين عن الاتصال بقاعدة البيانات عند تجاوز عدد المتصلين في نفس اللحظة الحد المسموح به، كما أنّ إجراء العمليّات عبر الشبكة أبطأ من إجرائها على جهاز المستخدم.. كل هذا جعل المبرمجين يستخدمون تقنية ADO بطريقة غريبة، فقد كانوا يتصلون بقاعدة البيانات وينسخون البيانات المطلوبة إلى أجهزتهم، ثمّ يغلقون الاتصال ويجرون العمليّات المطلوبة على البيانات على أجهزتهم.. وإذا كانت هناك تغييرات يجب حفظها في قاعدة البيانات، يتصلونَ بقاعدة البيانات مرّة أخرى ويحفظون البيانات ثمّ يغلقون الاتصال.

وقد أخذت ميكروسوفت هذا الأمر بعين الاعتبار، وطورت تقنية ADO مع ظهور فيجيوال ستديو دوت نت ٢٠٠٢، وصارت التقنية الجديدة تحمل الاسم ADO.NET، فصار بالإمكان التعامل مع البيانات بعد إغلاق الاتصال فيما عرف باسم "التعامل المنفصل" Disconnected Mode، حيث يتصل المستخدم بقاعدة البيانات ويقوم بتحميل البيانات منها إلى الذاكرة ويغلق الاتصال، ليتعامل مع هذه البيانات على جهازه، وعندما يريد حفظ التغييرات، يفتح الاتصال مرّة أخرى لنقل البيانات إلى قاعدة البيانات.

وفيما يلي تلخيص للخطوات التي تقوم بها عبر تقنية ADO.NET للحصول على البيانات وتحديثها:

- في البداية عليك أن تقوم بالاتصال بقاعدة البيانات بواسطة كائن الاتصال Connection object.. هذا الكائن يتيح لك توضيح اسم قاعدة البيانات وكيفية الاتصال بها.

- بعد هذا عليك استخدام كائن الأمر Command Object، الذي يتولى تنفيذ جملة الاستعلام SQL Query عبر الاتصال المفتوح.

- بعد هذا يكون أمامك أحد اختيارين:

١. فإما أن تستخدم قارئ البيانات Data Reader لقراءة نتائج الاستعلام مباشرة دون حفظها على جهاز العميل.

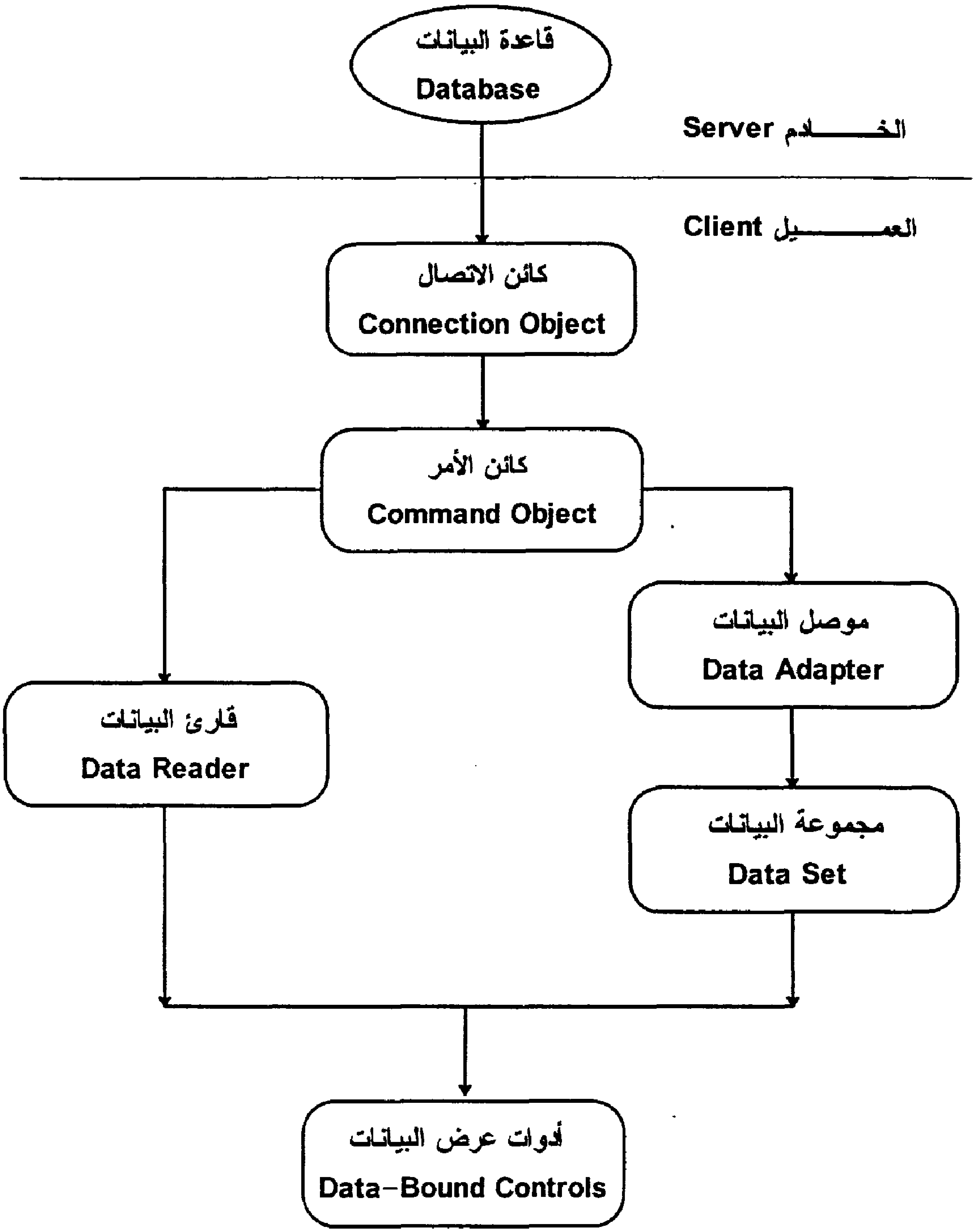
٢. وإما أن تستخدم "موصّل البيانات" Data Adapter لحفظ نتائج الاستعلام على جهاز العميل في مجموعة البيانات Data Set، التي يمكن القول إنّها صورة مصغّرة من قاعدة البيانات، تحتوي على الجداول والعلاقات Relations، والقيود Constraints المفروضة على قيم الحقول، بما في ذلك فرض التكامل المرجعي Referential Integrity بين الجداول.

- بعد هذا عليك إغلاق الاتصال، ومعالجة البيانات في برنامجك.. ويمكنك عرض البيانات للمستخدم لقراءتها أو تعديلها باستخدام أدوات ربط البيانات Data Bound Controls.

- وإذا كانت هناك أية تغييرات أجراها المستخدم على البيانات وتريد حفظها في قاعدة البيانات، فعليك استخدام كائن الاتصال مرة أخرى للاتصال بها، واستخدام كائن الأمر أو موصل البيانات لتنفيذ استعلام التحديث.

كما ترى: يعتمد هذا التنظيم على تقسيم العمل إلى طبقات Layers مستقلّة في وظيفتها، وبهذا يسهل عليك التعديل في أي طبقة دون هدم الطبقات السابقة أو التالية لها، مما يوفر الوقت والجهد.

والشكل التالي يلخّص هذه التقنية:




وتوجد فئات ADO.NET في نطاقات الأسماء Namespaces التالية:

- System.Data
- System.Transactions
- Microsoft.SqlServer.Server

## لغة XML:

الحروف XML هي اختصار للتعبير "لغة التوصيف القابلة للتمديد" Extensible Markup Language، وهي طريقة لتمثيل أيّ بيانات لها تركيب منظم، وذلك بتحويلها إلى نصّ يعبّر عنها.. لهذا تصلح لغة XML للتعبير عن أيّ نوع من أنواع البيانات، كالجداول والصور وغيرهما.

ورغم أنّ الملفّات النصيّة أكبر حجما من الملفات الثنائيّة Binary Files، إلا إنّ الأولى صالحة للتعامل مع أيّ برنامج بل مع أيّ نظام تشغيل، دون الوقوع في مشاكل اختلاف طرق تمثيل البيانات الثنائية.. لهذا صارت لغة XML في السنوات الأخيرة أنسب وسيلة لنقل البيانات عبر الإنترنت، لأنها تتجاوز مشاكل عدم التوافق بين التطبيقات وأنظمة التشغيل المختلفة.. ولهذا تستخدم تقنية ADO.NET لغة XML في نقل البيانات بين الخادم والعميل.

ورغم أن إطار العمل يقدم دعما كاملا للغة XML ويتيح لك كتابتها وقراءتها، إلا أنك لن تحتاج إلى هذا عند التعامل مع قواعد البيانات، لأنّ تقنية ADO.NET تستخدم لغة XML كطبقة داخليّة، فهي تنتجها وتقرأ البيانات منها بطريقة آليّة.

ومن الإمكانيّات التي تتيحها لك لغة XML، قدرتك على استخدامها لإنشاء مجموعة بيانات DataSet بدون الاعتماد على أيّ قاعدة بيانات.. في هذه الحالة ستوضع البيانات في الذاكرة، ولو شئت الاحتفاظ بها، فيمكنك حفظها في ملفّ، ثمّ إعادة تحميلها مرّة أخرى حينما تريد.. وسنتعرف على هذا بتفصيل اكبر لاحقا.

## مزودات قواعد البيانات Database Providers:

توفر تقنية ADO.NET عدة مزودات Providers للتعامل مع أنواع مختلفة من قواعد البيانات.. وهذه المزودات هي:

### ١- ODBC:

اسم هذا المزود هو اختصار للمصطلح "التواصل المفتوح مع قواعد البيانات":

Open Database Connectivity

وقد طورت ميكروسوفت هذه التقنية ـ بالتعاون مع آخرين، عام ١٩٩٢، لتوفر طريقة عامة للتعامل مع قواعد البيانات بغض النظر عن لغة البرمجة المستخدمة ونظام التشغيل الذي تعمل عليه، وتطبيق قواعد البيانات المستخدم.

وتوجد فئات هذا المزود في النطاق:

System.Data.ODBC

## ٢- OLE DB:

اسم هذا المزود هو اختصار للمصطلح "قاعدة بيانات ربط وتضمين الكائنات":

**O**bject **L**inking and **E**mbedding **D**ata**b**ase

وهو مزوّد Provider بنته ميكروسوفت باستخدام تقنية COM كتطوير وتحسين لتقنية ODBC، للتعامل بطريقة عامة مع أي نوع من أنواع قواعد البيانات، لهذا تستطيع استخدامه للتعامل مع آكسيس (فليس لقواعد بياناته مزود خاص بها)، وكذلك مع قواعد بيانات سيكويل سيرفر وأوراكل (رغم أن لكل منهما مزودا خاصا بهما)، ومع أي نوع آخر من أنواع قواعد البيانات، حتى ولو لم تكن تدعم استخدام لغة SQL مثل الجداول الشاملة Spreadsheets الخاصة بتطبيق إكسيل Excel.

وتوجد فئات هذا المزود في النطاق:

System.Data.OleDb

## ٣- SQL Server:

توفر دوت نت دعما خاصا لسيكويل سيرفر باعتباره أهم تطبيقات قواعد البيانات التي أنتجتها ميكروسوفت بعد انتشار استخدام الشبكات والإنترنت في عالم التجارة والأعمال.

وتوجد فئات هذا المزود في النطاقات التالية:

| | |
|---|---|
| System.Data.SqlClient | فئات مزود سيكويل سيرفر. |
| System.Data.SQL | يقدم بعض الوظائف الخاصة بسيكويل سيرفر. |
| System.Data.SqlTypes | يحتوي على فئات تمثل أنواع البيانات Data Types الخاصة بسيكويل سيرفر، ليمكنك استخدامها بدلا من أنواع البيانات الموجودة في إطار العمل |
| Microsoft.SqlServer.Server | يحتوي على الفئات اللازمة لتشغيل سيكويل سيرفر في دوت نت. |

٤- SQL Server Compact 3.5:

يتيح لك هذا المزود التعامل مع قواعد البيانات المنشأة بالنسخة الخفيفة من سكيويل سيرفر SQL Server Compact Edition، المخصصة لإنشاء قواعد بيانات للأجهزة الكفية المحمولة، التي تتعامل مع النسخة الخفيفة من الويندوز Windows CE والنسخة الخفيفة من إطار العمل .NET Compact Framework..

وتوجد فئات هذا المزود في النطاق:

System.Data.OracleClient

لكن استخدام هذا النطاق يتطلب منك أولا إضافة مرجع إليه في برنامجك، علما بأنه يوجد في الملف:

system.data.sqlserverce.dll

٥- Oracle:

قدمت ميكروسوفت منذ إصدار دوت نت ٢٠٠٣ دعما للتعامل مع قواعد بيانات أوراكل، فهي تمتاز بالقوة والشهرة والانتشار.

وتوجد فئات هذا المزود في النطاق: System.Data.OracleClient

لكن استخدام هذا النطاق يتطلب منك أولا إضافة مرجع إليه في برنامجك، علما بأنه يوجد في الملف:

System.Data.OracleClient.dll

وعليك أن تلاحظ أن جميع هذه المزودات توفر نفس أدوات الاتصال بقاعدة البيانات (كائن الاتصال Connection، كائن الأمر Command، موصل البيانات DataAdapter، مجموعة البيانات DataSet، قارئ البيانات DataReader... إلخ)، لكنّ كلا منها يبدأ باختصار يوضح نوع المزود، مثل:

| | |
|---|---|
| كائنات الاتصال | SqlCeConnection<br>SqlConnection<br>OdbcConnection<br>OleDbConnection<br>OracleConnection |
| كائنات الأمر | SqlCeCommand<br>SqlCommand<br>OdbcCommand<br>OleDbCommand<br>OracleCommand |

| | |
|---|---|
| SqlCeDataAdapter<br>SqlDataAdapter<br>OdbcDataAdapter<br>OleDbDataAdapter<br>OracleDataAdapter | موصلات البيانات |
| SqlDataSet<br>OdbcDataSet<br>OleDbDataSet<br>OracleDataSet | مجموعات البيانات |
| SqlCeDataReader<br>SqlDataReader<br>OdbcDataReader<br>OleDbDataReader<br>OracleDataReader | قارئات البيانات |

ونظرا لأنه لا توجد فروق تذكر بين أنواع الكائنات الخاصة بأحد المزودات والكائنات الخاصة بنوع آخر، فسنقتصر في هذا الكتاب على شرح مزود سيكويل سيرفر، لأن استخدامك لباقي أنواع المزودات لن يختلف في شيء، سوى في تغيير نطاق الاسم والبادئة التي تسبق اسم كل كائن من كائنات التعامل مع قاعدة البيانات!

-٦-

# أنواع بيانات سيكويل المدارة

# Managed SQL Data Types

يمنحك النطاق System.Data.SqlTypes عددا من السجلات Structures والفئات Classes التي تمثل أنواع البيانات الخاصة بخادم سيكيول.. هذا يسهل عليك إرسال واستقبال البيانات عند التعامل مع قواعد بيانات سيكيول.

وفيما يلي، نتعرف على هذه السجلات.. لا تنسَ إضافة الجملة التالية أعلى صفحة الكود، قبل تجربة أي مثال في هذا الفصل:

**Imports System.Data.SqlTypes**

وستجد أمثلة على بعض هذه الأنواع في المشروع SqlDataTypes.

لاحظ أن جميع الأنواع التي سنتعرف عليها تمثل الواجهة INullable، لهذا فهي تستطيع أن تحتوي القيمة Null، مما يعني أن الخانة التي يتعامل معها الكائن في قاعدة البيانات فارغة.. كما يعني أن جميع هذه الأنواع تمتلك الخاصية IsNull، التي تعيد True إذا كان الكائن فارغا (يحتوي على Null)، وفي هذه الحالة يجب ألا تحاول قراءة قيمة هذا الكائن، وإلا حدث خطأ في البرنامج.

## سجل القيمة المنطقية SqlBoolean Structure

يستطيع هذا السجل أن يحتوي على قيمة منطقية: True أو False.

ولحدث إنشاء هذا السجل ثلاث صيغ:

١. الأولى بدون معاملات، وهي تنشئ نسخة قيمتها Null.

٢. والثانية تستقبل قيمة منطقية Boolean لوضعها في السجل.. مثال:

**Dim Sb As New SqlBoolean(True)**

٣. والثالثة تستقبل عددا صحيحا Integer لوضعه في هذا السجل، حيث يعتبر الصفر False وأي عدد آخر True.

ويمتلك هذا السجل الخصائص التالية:

**S 📄 خطأ False:**

تعيد سجلا منطقيا SqlBoolean قيمته False.

**S 📄 صواب True:**

تعيد سجلا منطقيا SqlBoolean قيمته True.

**S 📄 عدم Null:**

تعيد سجلا منطقيا SqlBoolean قيمته Null.

**S 📄 صفر Zero:**

تعيد سجلا منطقيا SqlBoolean قيمته ٠ (هذا يعني أن قيمته False).

**S 📄 واحد One:**

تعيد سجلا منطقيا SqlBoolean قيمته ١ (هذا يعني أن قيمته True).

**هل هو خطأ IsFalse:**

تعيد True إذا كانت قيمة السجل الحالي False.

**هل هو صواب IsTrue:**

تعيد True إذا كانت قيمة السجل الحالي True.

**القيمة Value:**

تعيد قيمة منطقية Boolean تعبر عن قيمة السجل الحالي.. وتسبب هذه الخاصية خطأ إذا كان السجل منعدما، لهذا عليك فحصه أولا باستخدام الخاصية IsNull قبل استخدام هذه الخاصية.

### القيمة الرقمية ByteValue:

تعيد وحدة ثنائية Byte تعبر عن قيمة السجل الحالي (صفر للخطأ و ١ للصواب).. وتسبب هذه الخاصية خطأ إذا كان السجل منعدما، لهذا عليك فحصه أولا باستخدام الخاصية IsNull قبل استخدام هذه الخاصية.. مثال:

```
If Not Sb.IsNull Then
  MsgBox(Sb.ByteValue) ' 1
  MsgBox(Sb.Value) ' True
End If
```

ويمتلك هذا السجل المعاملات Operators اللازمة لإجراء العمليات الحسابية والمنطقية اللازمة.. كما أنه يمتلك وسائل مشتركة Shared Methods لأداء نفس وظائف هذه المعاملات.. والجدول التالي يلخص المعاملات المتاحة والدوال المناظرة لها:

| المعامل | الوسيلة |
|---|---|
| And | And |
| Or | Or |
| Xor | Xor |
| Not | OnesComplement (المعكوس الثنائي) |
| = | Equals |
| <> | NotEquals |
| > | GreaterThan |
| >= | GreaterThanOrEquals |
| < | LessThan |
| <= | LessThanOrEquals |

كما يمتلك هذا السجل عدة وسائل للتحويل، مثل:

- **ToString**
- **ToSqlByte**
- **ToSqlDecimal**
- **ToSqlDouble**
- **ToSqlInt16**
- **ToSqlInt32**
- **ToSqlInt64**
- **ToSqlMoney**
- **ToSqlSingle**
- **ToSqlString**
- **Parse** S

انظر المثال التالي:

```
Dim Sb = SqlBoolean.Parse("false")
Dim B As SqlByte = Sb.ToSqlByte
MsgBox(B.ToString) ' 0
```

مع ملاحظة أنك لا تحتاج إلى استخدام هذه الوسائل، لأن هذا السجل يعرف أيضا معاملات التحويل الضمني Implicit Operators ومعاملات التحويل الصريح Explicit Operators اللازمة لتحويل القيم الأخرى إلى هذا السجل، أو تحويل هذا السجل إلى قيم أخرى مباشرة.. مثال:

```
Dim Sb1 As SqlBoolean = "true"
Dim B1 As SqlByte = Sb1
MsgBox(B1.ToString) ' 1
```

| **ملحوظة:** |
| --- |
| كل الأنواع التي سنشرحها فيما بعد مزودة بالمعاملات الحسابية (الطرح والجمع والضرب والقسمة وباقي القسمة) والمعاملات المنطقية (And و Or و Xor و Not) ومعاملات التحويل الضمني والصريح.. ولا يحتوي كل نوع إلا على المعاملات التي تناسب القيم الموجودة فيه (النصوص مثلا لا تملك معاملات منطقية)، لهذا لن نكرر ذكر هذا في باقي الأنواع، إلا إذا كان هناك معامل يقوم بوظيفة مختلفة عن المألوف. |

## سجل الوحدة الثنائية SqlByte Structure

هذا السجل يحفظ وحدة ثنائية بدون إشارة، أي أنه يتعامل مع الأعداد من ٠ إلى ٢٥٥.
ويستقبل حدث إنشاء هذا السجل وحدة ثنائية Byte لنسخ قيمتها إليه.. مثال:

```
Dim B As New SqlByte (5)
```

ويمتلك هذا السجل الخصائص التالية:

**أقل قيمة MinValue:**
تعيد أقل قيمة يمكن وضعها في السجل.

**أقصى قيمة MaxValue:**
تعيد أكبر قيمة يمكن وضعها في السجل.

**صفر Zero:**
تعيد نسخة من السجل SqlByte تحتوي على القيمة صفر.

**العدم Null:**
تعيد نسخة من السجل SqlByte لا تحتوي على أي قيمة.

**القيمة Value:**
تعيد وحدة ثنائية Byte تحمل القيمة المحفوظة في السجل.. وتسبب هذه الخاصية خطأ في البرنامج إذا كان السجل منعدما، لذا عليك أن تستخدم الخاصية IsNull أولا للتأكد من وجود قيمة في السجل.

لاحظ أن ما ينطبق على السجل SqlByte ينطبق على السجلات الرقمية الأخرى، فهي تمتلك نفس الخصائص، وتستطيع حفظ قيمة أو Null، والاختلاف الوحيد هو نوع القيمة المحفوظة.. لذا فلا داعي لتكرار نفس الكلام مع الأنواع التالية، فأنت تستطيع فهمها بمجرد النظر:

- سجل الأعداد القصيرة SqlInt16 Structure
- سجل الأعداد الصحيحة SqlInt32 Structure
- سجل الأعداد الطويلة SqlInt64 Structure
- سجل الأعداد المفردة SqlSingle Structure
- سجل النقود SqlMoney Structure
- سجل الأعداد المزدوجة SqlDouble Structure
- سجل التاريخ والوقت SqlDateTime Structure
- سجل المعرف المتفرد العام SqlGuid Structure

## سجل الأعداد العشرية SqlDecimal Structure

يحفظ هذا السجل الأعداد العشرية، بنفس طريقة السجل Decimal، كما أن حدث إنشاء هذا السجل يمتلك صيغا شبيهة بحدث إنشاء السجل Decimal، التي يمكنك مراجعتها في كتاب برمجة إطار العمل.
ويزيد هذا السجل على سجلات الأنواع العددية الأخرى بالخصائص التالية:

**أقصى دقة MaxPrecision:**
يعيد أقصى عدد من الخانات الصحيحة والعشرية يمكن استخدامه في العدد.

**أقصى مقياس MaxScale:**
يعيد أقصى عدد من الخانات العشرية يمكن استخدامه في العدد.

**البيانات الثنائية BinData:**
تعيد مصفوفة ثنائية Byte Array تحتوي على التمثيل الثنائي للعدد العشري.

**البيانات Data:**
تعيد مصفوفة أعداد صحيحة Integer Array تحتوي على التمثيل الثنائي للعدد العشري.

**هل هو موجب IsPositive:**

تعيد True إذا كان العدد العشري موجبا.

**الدقة Precision:**

تعيد عدد الخانات الصحيحة والعشرية في العدد الحالي.

**المقياس Scale:**

تعيد عدد الخانات العشرية في العدد الحالي.

## فئة الحروف SqlChars Class

هذه الفئة تتعامل مع مصفوفة حروف، بحيث يمكن استخدامها للتعامل مع أنــواع ســيكيول التاليــة: varchar, nvarchar, char, nchar, text, ntext ، مــع ملاحظة أن أقصى عدد من الحروف يمكن وضعه في هذه الفئة هو أقصى قيمــة للعدد الصحيح (أي حوالي ٢ مليار حرف).

ولحدث إنشاء هذه الفئة ثلاث صيغ:

١. الأولى بدون معاملات، وهي تنشئ نسخة قيمتها Null.
٢. والثانية تستقبل مصفوفة حروف Char Array.
٣. والثالثة تستقبل نسخة من السجل SqlString لأخذ الحروف من الـنص الموجود فيها.

وتمتلك هذه الفئة الخصائص التالية:

**العدم Null:**

تعيد نسخة فارغة من الفئة SqlChars.

**العنصر Item:**

هذه هي الخاصية الافتراضية Default Property، وهي تقــرأ أو تغيــر الحرف الموجود في الموضع المرسل كمعامل.

**الطول Length:**

تعيد عدد الحروف الموجودة حاليا في الكائن.

### أقصى طول MaxLength:
تعيد أقصى عدد من الحروف يمكن وضعه في الكائن.. هذا العدد يساوي صفرا مبدئيا، لكنه يساوي طول النص عند وضع نص في الكائن.. وعند تقصير طول الكائن، يظل أقصى طول كما هو دون أن ينقص.

### التخزين Storage:
تعيد إحدى قيم المرقم StorageState التي توضح نوع المخزن الذي يتم فيه حفظ الحروف داخل الكائن، وهذه القيم هي:

| | |
|---|---|
| Buffer | تحفظ الحروف في مصفوفة داخلية. |
| UnmanagedBuffer | تحفظ الحروف في الذاكرة باستخدام مؤشرات غير مدارة بإطار العمل. |
| Stream | تحفظ الحروف في مجرى بيانات Stream. |

### المخزن الوسيط Buffer:
تعيد مصفوفة الحروف التي يتعامل معها الكائن داخليا.. لاحظ أن أي تغيير في هذه المصفوفة يؤثر على محتويات الكائن.

### القيمة Value:
تعيد مصفوفة حروف بها نسخة من محتويات الكائن.. لاحظ أن أي تغيير في هذه المصفوفة لا يؤثر على محتويات الكائن، على عكس المصفوفة التي تعيدها الخاصية Buffer.

كما تمتلك فئة الحروف الوسائل التالية:

### تغيير الطول SetLength:
أرسل إلى هذه الخاصية عدد الحروف الذي تريد وجودها في الكائن.. لاحظ أنك لو أرسلت عددا أكبر من أقصى طول MaxLength فسيحدث خطأ.. هذا معناه أنك تستطيع تصغير محتويات الكائن، حيث سيتم حذف الحروف الزائدة عن الطول الجديد، لكن ستظل المصفوفة الداخلية تحجز الخانات التي تم الاستغناء عنها، لهذا تستطيع أن تكبر الطول مرة أخرى، بشرط عدم تجاوز الطول الأقصى.

ولو كان الكائن يتعامل مع مخزن وسيط غير مدار Unmanaged Buffer فسيتم تحويله إلى مخزن مدار Managed Buffer بعد تنفيذ هذه الوسيلة.

### وضع العدم SetNull:
تمحو محتويات الكائن الحالي وتجعل طوله صفرا.

### قراءة Read:

تنسخ عددا من الحروف من الكائن الحالي إلى مصفوفة، ولها المعاملات التالية:

- موضع بداية القراءة من الكائن.
- مصفوفة الحروف التي سيتم النسخ إليها.
- موضع بداية الكتابة في المصفوفة.
- عدد الحروف المنسوخة.

وتعيد هذه الوسيلة عدد الحروف التي تم نسخها.. الحكمة في هذا أن عدد الحروف المنسوخة قد يكون أقل من المطلوب، إذا لم يكن الكائن يحتوي على العدد المطلوب من الحروف.

والمثال التالي ينسخ ٥ حروف من الكائن بدءا من الحرف الرابع:

```
Dim Sc As New SqlChars("This is a test")
Dim C(4) As Char
Sc.Read(3, C, 0, 5)
MsgBox("'" & C & "'")
```

### كتابة Write:

تنسخ عددا من الحروف من مصفوفة إلى الكائن الحالي بدءا من موضع معين.. ولها نفس معاملات الوسيلة السابقة.

لاحظ أنك تستطيع كتابة حروف في موضع تال لآخر حرف في الكائن، بشرط ألا تتجاوز الطول الأقصى للكائن MaxLength.

والمثال التالي ينسخ كل حروف المصفوفة إلى الكائن بدءا من الحرف الرابع:

```
Dim Sc As New SqlChars("This is a test")
Dim X() As Char = {"A"c, "B"c, "C"c, "D"c, "E"c}
Sc.Write(3, X, 0, 5)
MsgBox(Sc.Value)
```

### التحويل إلى نص سيكويل ToSqlString:

تعيد نسخة من السجل SqlString تحتوي على نص مكون من حروف الكائن الحالي.

## سجل النص SqlString Structure

يختلف نص سيكويل في طريقة تمثيله الداخلية، عن فئة النص String Class العادية.. فعلى سبيل المثال: يأخذ النص العادي معلومات الثقافة من اللغة الافتراضية المعرفة على جهاز المستخدم، بينما لا يفعل نص سيكويل هذا، فلو لم تمده بمعرف الثقافة، فإنه يستخدم مقاييس داخلية خاصة به لمقارنة النصوص.. ولو حاولت مقارنة نسختين من نص سيكويل لكل منهما معرف ثقافة LCID مختلف عن الآخر، فإن خطأ سيحدث في البرنامج بسبب عدم قدرته على إجراء عملية المقارنة.

(أنصح بمراجعة فصل العولمة Globalization في مرجع برمجة إطار العمل).

ولحدث إنشاء هذا السجل الصيغ التالية:

١. الأولى بدون معاملات، وهي تنشئ نسخة قيمتها Null.

٢. والثانية تستقبل نصا String لنسخه إلى السجل.

٣. والثالثة تزيد على الصيغة السابقة بمعامل ثان يستقبل معرف الثقافة LCID الذي تريد استخدامه عند مقارنة النص بأي نص آخر.. مثال:

**Dim Ss As New SqlString("محمد", System.Globalization.
CultureInfo.CurrentCulture.LCID)**

٤. والرابعة تزيد على الصيغة السابقة بمعامل ثالث، يستقبل إحدى قيم المرقم SqlCompareOptions التالية:

| | |
|---|---|
| None | تتم المقارنة بالخيارات الافتراضية للثقافة التي يرتبط بها السجل الحالي. |
| IgnoreCase | تتجاهل المقارنة حالة الأحرف. |
| IgnoreNonSpace | تتجاهل المقارنة كل الرموز التي لا تعتبر فواصل بين الحروف، مثل علامات التشكيل في اللغة العربية.. هذا مفيد عند البحث عن كلمة مع تجاهل التشكيل. |
| IgnoreKanaType | تتجاهل المقارنة الرموز الصوتية في اللغة اليابانية. |
| IgnoreWidth | تتجاهل المقارنة إن كانت الحروف اليابانية مكتوبة بالعرض الكامل أم بنصف العرض. |

| | |
|---|---|
| BinarySort | يتم ترتيب الحروف تبعا لقيمها الرقمية في ترميز ASCII أو Unicode، وليس تبعا للترتيب الهجائي. |
| BinarySort2 | يتم ترتيب الحروف ثنائيا، باستخدام قيمها في الترميز. |

لاحظ أنك تستطيع دمج أكثر من قيمة معا باستخدام المعامل Or.

٥. والخامسة تستقبل معرفة الثقافة LCID وخيارات المقارنة SqlCompareOptions ومصفوفة ثنائية Byte Array تحتوي على التمثيل الثنائي للنص.
٦. والسادسة تزيد على الصيغة السابقة بمعامل رابع، عليك جعله True إذا كان النص ممثلا بالترميز الموسع Unicode.
٧. والسابعة تزيد على الصيغة الخامسة بمعامل رابع يستقبل موضع بداية القراءة من المصفوفة، ومعامل خامس يستقبل عدد الحروف التي تريد قراءتها منها.
٨. والثامنة تزيد على الصيغة السابقة بمعامل رابع، عليك جعله True إذا كان النص ممثلا بالترميز الموسع Unicode.

ويمتلك سجل النص الخصائص التالية:

**ترتيب ثنائي BinarySort:**

تعمل كثابت يعني أن ترتيب الحروف يتم تبعا لقيمها الرقمية في ترميز ASCII وليس تبعا للترتيب الهجائي.

**ترتيب ثنائي BinarySort2:**

تعمل كثابت يعني أن ترتيب الحروف يتم تبعا لقيمها الرقمية في الترميز.

**تجاهل الحالة IgnoreCase:**

تعمل كثابت يعني أن مقارنة الحروف تتجاهل حالتها (صغيرة أم كبيرة).

**تجاهل الحروف غير الفاصلة IgnoreNonSpace:**

تعمل كثابت يعني أن المقارنة تتجاهل كل الرموز التي لا تعتبر فواصل بين الحروف، مثل علامات التشكيل في اللغة العربية.

**S 🔒 تجاهل نوع الكانا IgnoreKanaType:**

تعمل كثابت يعني أن مقارنة الحروف تتجاهل الرموز الصوتية في اللغة اليابانية.

**S 🔒 تجاهل العرض IgnoreWidth:**

تعمل كثابت يعني أن مقارنة الحروف تتجاهل إن كانت الحروف اليابانية مكتوبة بالعرض الكامل أم بنصف العرض.

**S 🔒 العدم Null:**

تعيد نسخة فارغة من السجل SqlString.

**🔒 معلومات الثقافة CultureInfo:**

تعيد كائن معلومات الثقافة CultureInfo الذي يستخدمه السجل الحالي في تنسيق وترتيب ومقارنة النصوص.

**🔒 المعرف المحلي للثقافة LCID:**

تعيد عددا صحيحا يستخدم كمعرف للثقافة التي يتعامل معها السجل الحالي.

**🔒 معلومات المقارنة CompareInfo:**

تعيد كائن معلومات المقارنة CompareInfo الذي يستخدمه السجل الحالي في مقارنة النصوص.

**🔒 خيارات المقارنة SqlCompareOptions:**

تعيد إحدى قيم المرقم SqlCompareOptions التي توضح الخيارات المستخدمة لمقارنة النصوص.

**🔒 القيمة Value:**

تعيد String يمثل النص الموجود في السجل الحالي، أو تسبب خطأ إذا كان السجل فارغا.

ويمتلك هذا السجل الوسائل الهامة التالية:

### تشبيك Concat:

تدمج نسختين من نص سيكيول في نص واحد، وتعيده كسجل جديد.. مثال:

```
Dim LCID = System.Globalization.
        CultureInfo.CurrentCulture.LCID)
Dim Ss1 As New SqlString("محمد", LCID)
Dim Ss2 As New SqlString(" محمود", LCID)
Dim Ss3 = SqlString.Concat(Ss1, Ss2)
MsgBox(Ss3.Value) ' محمد محمود
```

ويمكنك إنجاز نفس المهمة باستخدام الوسيلة Add كالتالي:

```
Dim Ss3 = SqlString.Add(Ss1, Ss2)
```

أو باستخدام معامل الجمع كالتالي:

```
Dim Ss3 = Ss1 + Ss2
```

### خيارات المقارنة من خيارات مقارنة سيكويل :CompareOptionsFromSqlCompareOptions

أرسل إلى هذه الوسيلة إحدى قيم المرقم SqlCompareOptions، لتعيد إليك القيمة المناظرة لها في المرقم CompareOptions الذي تعرفنا عليه في كتاب برمجة إطار العمل.

### نسخ Clone:

تعيد نسخة جديدة من السجل SqlString بها نفس قيمة السجل الحالي.

### قراءة البيانات غير الموسعة GetNonUnicodeBytes:

تعيد مصفوفة ثنائية Bytes Array، تحتوي على تمثيل النص الحالي في ترميز ASCII.

### قراءة البيانات الموسعة GetUnicodeBytes:

تعيد مصفوفة ثنائية Bytes Array، تحتوي على تمثيل النص الحالي في ترميز Unicode.

## سجل البيانات الثنائية SqlBinary Structure

يمثل هذا السجل مصفوفة من الوحدات الثنائية Byte Array.. ويمكنك ملء هذا السجل بالبيانات بإرسال مصفوفة ثنائية إلى حدث إنشائه.. مثال:

```
Dim Sb1 As New SqlBinary({100, 220, 3})
```

ونظرا لأن هذا السجل يعرف معامل التحويل الضمني Implicit Operator، فيمكنك وضع مصفوفة ثنائية في هذا السجل مباشرة:

```
Dim Sb2 As SqlBinary = {1, 0, 2}
```

ويمتلك هذا السجل الخصائص التالية:

### القيمة المنعدمة Null:

تعيد سجلا فارغا.. مثال:

```
Dim Sb3 = SqlBinary.Null
```

### الطول Length:

تعيد عدد الوحدات الثنائية الموجودة في السجل.. مثال:

```
MsgBox(Sb1.Length) ' 3
```

### العنصر Item:

هذه هي الخاصية الافتراضية للسجل Default Property، وهي تعيد الوحدة الثنائية Byte الموجود في موضع معين في السجل.. مثال:

```
MsgBox(Sb1.Item(0)) ' 100
MsgBox(Sb1(1)) ' 220
```

المؤسف أن هذه الخاصية للقراءة فقط، لذا فلا يمكنك استخدامها لتغيير العنصر الموجود في موضع معين من المصفوفة.. ولست أدرى ما الحكمة من هذا!

### القيمة Value:

تعيد مصفوفة ثنائية Byte Array تحتوي على القيم الموجودة في السجل الحالي.

كما يمتلك هذا السجل عددا من الوسائل المشتركة Shared، وهي تقــوم بــنفس وظائف المعاملات Operators المعرفة لهذا السجل.. ويهمنا هنا أن نشير إلــى بعضها لأن طريقة عملها مختلفة نوعا:

**إضافة Add:**

**تشبيك Concat:**

**المعامل + :**

تقوم بتشبيك سجلين، بدمج المصفوفة الثانية بعد نهاية المصفوفة الأولى، وتعيد المصفوفة الجديدة في سجل جديد.. مثال:

```
Dim Sb = SqlBinary.Add(Sb1, Sb2)
```

أو:

```
Dim Sb = SqlBinary.Concat(Sb1, Sb2)
```

أو باختصار:

```
Dim Sb = Sb1 + Sb2
```

بعد تنفيذ هذا المثال، سيحتوي سجل البيانات الثنائية Sb على القيم التالية: ١٠٠، ٢٢٠، ٣، ١، ٠، ٢.

**يساوي Equals:**

**المعامل = :**

تعيد True إذا كان السجلان لهما نفس الطول ويحتويان على نفس القــيم بنفس الترتيب.. مثال:

```
MsgBox(SqlBinary.Equals(Sb1,Sb2)) ' False
```

أو باختصار:

```
MsgBox(Sb1 = Sb2) ' False
```

لاحظ أن القيمة العائدة من هذه الوسيلة هي من النــوع SqlBollean، حيــث تكون نتيجة المقارنة Null إذا كان أي من السجلين منعدما.

## فئة الوحدات الثنائية SqlBytes Class

هذه الفئة مشابهة بدرجة كبيرة للسجل SqlBinary، إلا أنها تمتلك ميزة إضافية، وهي قدرتها على التعامل مع الوحدات الثنائية Bytes من خلال مجرى بيانات Stream.. هذا يتيح لك قراءة البيانات من ملف FileStream أو من مجرى بيانات الذاكرة MemoryStream أو من خلال الشبكة NetworkStream.

ولحدث إنشاء هذه الفئة الصيغ التالية:

١. الصيغة الأولى بدون معاملات.
٢. والثانية تستقبل بياناتها من مصفوفة ثنائية Byte Array.
٣. والثالثة تستقبل بياناتها من سجل بيانات ثنائية SqlBinary.
٤. والرابعة تستقبل بياناتها من مجرى بيانات Stream.

وتشبه هذه الفئة أيضا الفئة SqlChars في كل خصائصها ووسائلها، ما عدا أن التعامل هنا يكون مع مصفوفة ثنائية Byte Array بدلا من مصفوفة حروف Char Array.. لهذا لا نحتاج إلى إعادة شرح هذه الخصائص والوسائل:

- العدم Null
- هل هو منعدم IsNull
- العنصر Item
- الطول Length
- أقصى طول MaxLength
- التخزين Storage
- المخزن الوسيط Buffer
- القيمة Value
- تغيير الطول SetLength
- وضع عدم SetNull
- قراءة Read
- كتابة Write

الجديد فقط، هو الخاصية والوسيلة التاليتان:

**مجرى البيانات Stream:**

تقرأ أو تغير مجرى البيانات الذي يتعامل معه الكائن الحالي.. ويؤدي استخدام هذه الخاصية إلى تحميل كل البيانات من مجرى البيانات إلى الذاكرة، ولو كانت هذه البيانات ضخمة للغاية، فقد تؤدي إلى استهلاك مساحة الذاكرة وحدوث خطأ من النوع OutOfMemoryException.

**التحويل إلى بيانات ثنائية ToSqlBinary:**

تعيد سجل بيانات ثنائية SqlBinary يحتوي على الوحدات الثنائية Bytes الموجودة في الكائن الحالي.

# فئة XML
# SqlXml Class

تحفظ هذه الفئة وثيقة XML، وهي تعتمد داخليا على "قـــارئ بيانـــات XML" XmlReader، لهذا يجب مراعاة أن يكون تنسيق بيانات XML الذي تـضعها في هذه الفئة موافقا للمعايير التي تقبلها الفئة XmlReader.. وسـنتعرف علـى كيفية التعامل مع بيانات XML وفئاتها في كتاب مستقل بإذن الله.

ولحدث إنشاء هذه الفئة ثلاث صيغ:

١- الأولى بدون معاملات.

٢- والثانية تستقبل البيانات من مجرى بيانات Stream.

٣- والثالثة تستقبل البيانات من "قارئ XML" XmlReader.

وتمتلك هذه الفئة الخصائص التالية:

### العدم Null:

تعيد نسخة من الفئة SqlXml لا تحتوي على أي قيمة.

### القيمة Value:

تعيد نصا String يحتوي على وثيقة XML المحفوظة في الكائن الحالي.

كما تمتلك هذه الفئة الوسيلة التالية:

### إنشاء قارئ CreateReader:

تعيد "قارئ XML" XmlReader لاستخدامه في قراءة محتويات الكـــائن الحالي.

| ملحوظة: |
|---|
| كل أنواع بيانات سيكويل السابقة تدعم التعامل مع XML، من خلال:<br>- تمثيل الواجهة IXmlSerializable لحفظ محتويات الكائن فـــي وثيقـــة XML وقراءتها منها في أي وقت.<br>- امتلاك وســيلة مــشتركة Shared Method اســمها GetXsdType، تستقبل "نوع مخطط XML" XmlSchemaSet، وتعيد نسخة من الفئة XmlQualifiedName تحتوي علـــى الاســم الكامــل لنـــوع XML المناظر. |

**حفظ الملفات خارج قاعدة البيانات:**

يقدم لك سيكيول سيرفر ٢٠٠٨ إمكانية رائعة، وهي قدرتك على حفظ البيانات الثنائية الضخمة BLOB (تكون في الغالب أكبر من ١ ميجا بايت) التي ترسلها إلى عمود من النوع varbinary(MAX) في ملف خاص مستقل عن ملف قاعدة البيانات، يتم حفظه في مجلد خاص على الخادم.. هذا يحقق لك الفوائد التالية:

١- يضمن عدم تضخم حجم ملف قاعدة البيانات بصورة كبيرة.

٢- يقلل من الزمن اللازم لقراءة هذه البيانات.

٣- يستطيع النوع varbinary(MAX) حفظ بيانات حجمها تقريبا ٢ جيجا بايت في قاعدة البيانات، بينما عند استخدام ملفات خارجية لا يكون هناك اي حد لجم الملف، إلا مقدار المساحة المتوفرة على القرص الصلب!

٤- قدرتك على التعامل مع هذه الملفات من خلال استعلامات قاعدة البيانات، أو التعامل معها مباشرة من خلال نظام ملفات الويندوز Windows File System.

٥- تقدم لك دوت ت ٢٠١٠ فئة خاصة للتعامل مع البيانات المحفوظة خارج قاعدة البيانات، وهي الفئة SqlFileStream التي سنتعرف عليها لاحقا.

وهناك أربع خطوات عليك اتباعها، حتى تستطيع حفظ قيم الأعمدة الضخمة في ملفات مستقلة.. هذه الخطوات هي:

**١- تفعيل استخدام مجرى البيانات FILESTREAM في خدمات الويندوز:**

يتم هذا كما يلي:

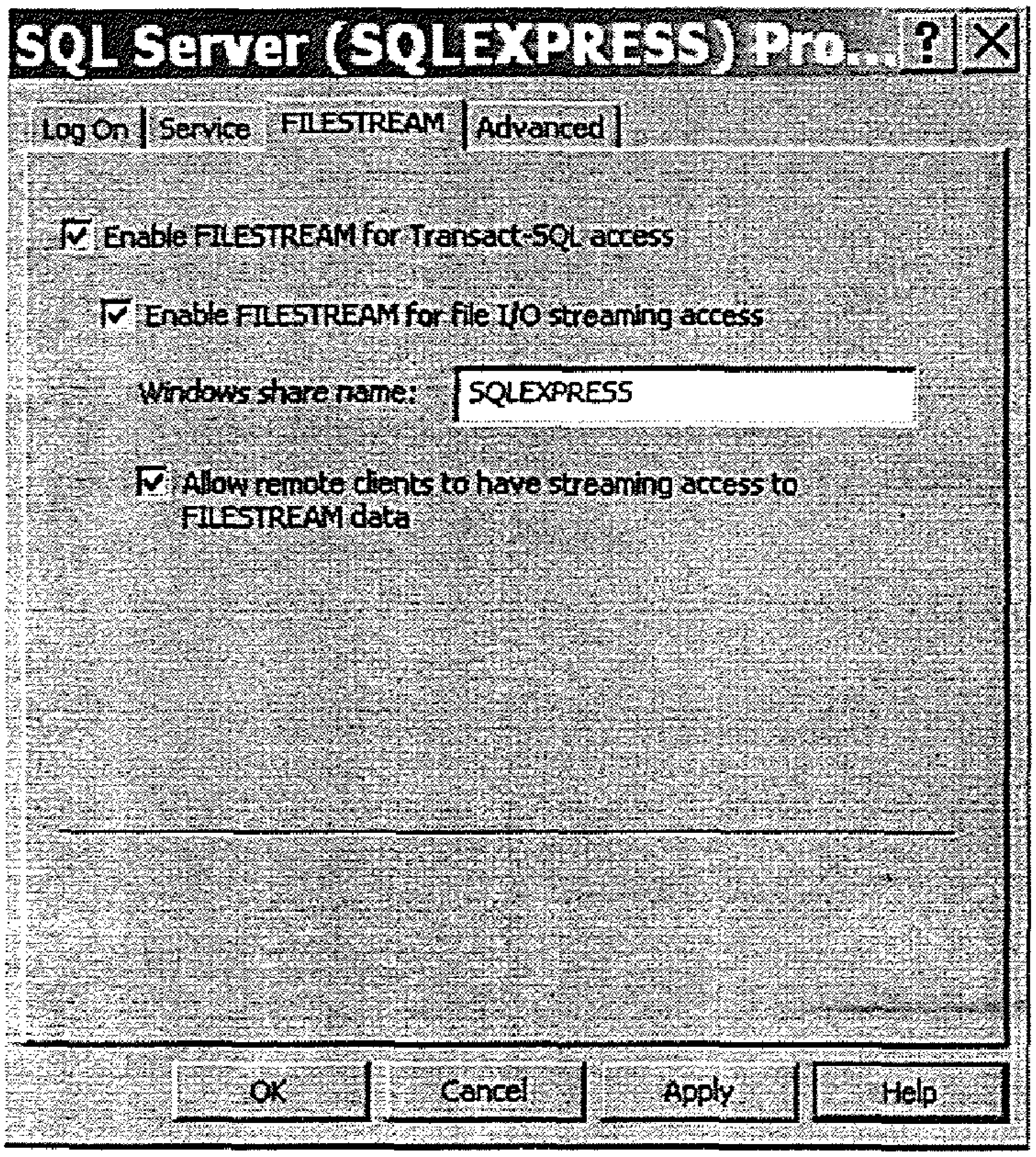
- من قائمة البرامج Programs Menu، اضغط:

Microsoft SQL Server 2008\Configuration Tools\
SQL Server Configuration Manager

- في الشجرة اليسرى في نافذة تهيئة خادم سيكويل، انقر العنصر SQL Server Services مرتين بالفأرة.
- في القائمة اليمنى، حدد اسم خادم سيكويل الذي تتعامل معه.. في حالتنا هذه سيكون SQL Server(SQLEXPRESS)، وانقره مرتين بالفأرة لعرض خصائصه.
- في نافذة الخصائص، اضغط الشريط العلوي Tab المسمى FILESTREAM لعرض صفحة خصائصه.
- ضع علامة الاختيار أمام الاختيار:

Enable FILESTREAM for Transact-SQL access

هذا سيتيح لك قراءة وكتابة الملفات الخارجية من خلال الاستعلامات.




- إذا وضعت علامة الاختيار أمام الاختيار:

Enable FILESTREAM for file I/O streaming access

فسيتيح هذا لك القراءة من الملفات الخارجية من خلال نظام مشاركة الملفات Sharing عبر شبكات الويندوز.. أي أنك ستتطيع التعامل مع الملف مباشرة بدون استعلامات، كأنك تتعامل مع أي ملف عادي على الشبكة، وهو ما سنفعله باستخدام الفئة SqlFileStream.. ويجب عليك أن تكتب في مربع النص اسم مجلد المشاركة الذي ستقرأ الملف من خلاله.. في الوضع الافتراضي يكون هذا الاسم هو SQLEXPRESS، لكن يمكنك تغييره إلى ما تشاء.

- إذا وضعت علامة الاختيار أمام الاختيار:

Allow remote clients to have streaming access to FILESTREAM

فسيسمح هذا للمستخدمين من خارج الشبكة المحلية Remote Users بقراءة بيانات الملف عبر نظام مشاركة الملفات.

**ملحوظة هامة:**

يشكو كثير من المستخدمين عند استخدام الفئة SqlFileStream من أن رسالة خطأ تظهر لهم تخبرهم بأن مسار الملف غير موجود على الشبكة.. يعود هذا السبب في الغالب إلى تعطيلهم لإمكانية مشاركة الملفات Sharing الخاصة بالويندوز، لهذا عليك التأكد من تفعيلها قبل تفعيل الاختيار:

Enable FILESTREAM for file I/O streaming access

ولفعل هذا، اتبع الخطوات التالية:

- افتح متصفح الويندوز Windows Explorer ومن القائمة الرئيسية Tools اضغط Folder Options.
- في نافذة خيارات المجلدات، اضغط الشريط العلوي View، وتأكد من وضع علامة الاختيار أمام الخيار الأخير في قائمة الخيارات:

Use simple file sharing.

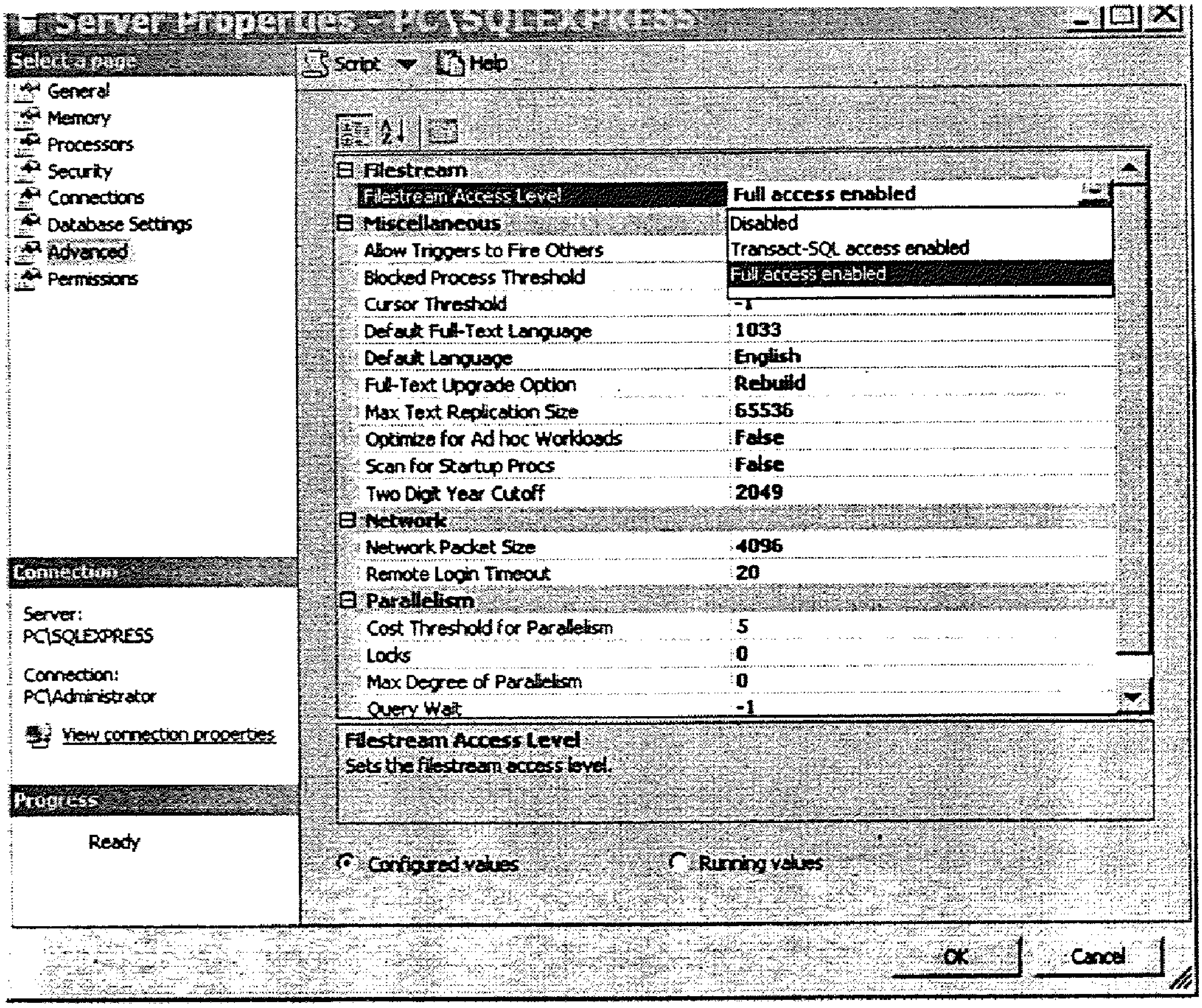
- اضغط Ok لإغلاق النافذة.

وإذا كنت فعّلت خيارات مجرى البيانات FileStream الخاصة بخادم سيكويل قبل تفعيل المشاركة، فقم بتعطيل الخيارات FileStream، وأعد تشغيل خادم سيكويل Restart، ثم أعد تفيعل خيارات مجرى البيانات.. بهذه الطريقة ستضمن وصول برنامجك إلى ملفات المشاركة الخاصة بسيكويل سيرفر، والتي يحفظ فيها الملفات الخارجية.

- اضغط Ok لإغلاق النافذة وحفظ هذه التغييرات.

**٢- تفعيل استخدام مجرى البيانات FILESTREAM في خادم سيكيول:**

لفعل هذا، افتح مدير سيكويل SQL Server Management Studio، وفي متصفح الكائنات Object Browser حدد العنصر الرئيسي في الشجرة (الذي يحمل اسم خدادم سيكويل SQLEXPRESS)، واضغطه بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Properties.. سيعرض هذا نافذة خصائص خادم سيكويل.. اضغط العنصر Advanced من القائمة اليسرى، لعرض الخصائص المتقدمة، كما هو موضح في الصورة:



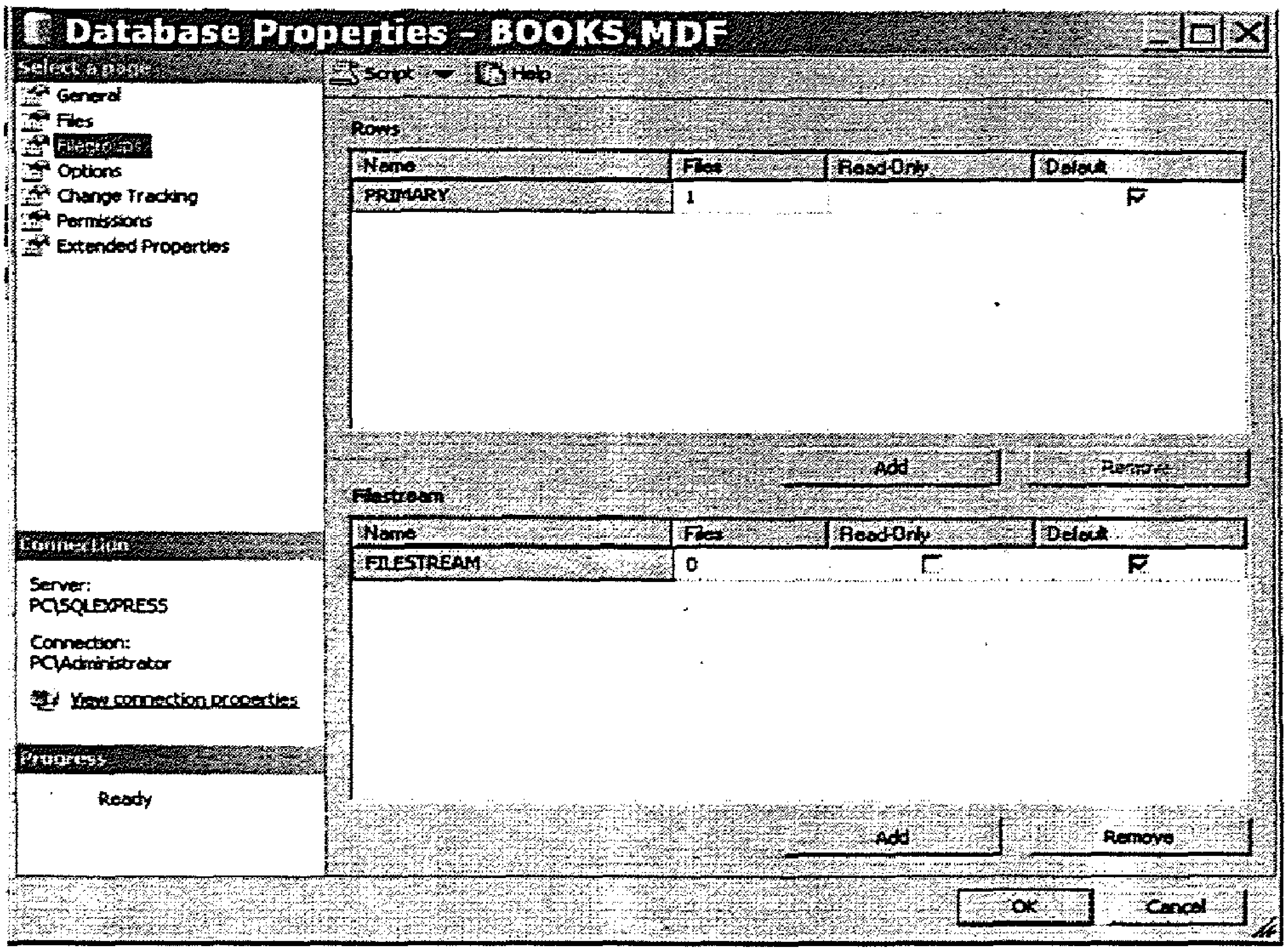

ستجد أول خاصية فيها هي Filestream Access Level، التي توضح كيفية التعامل مع الملفات الخارجية.. ستجد قيمة هذه الخاصية Disabled أي أن التعامل مع الملفات الخارجية ممنوع!.. اضغط القائمة المنسدلة، واختر التعامل الكامل Full Access Enabled.. هذا يسمح بالتعامل مع الملفات مباشرة، أو من خلال الاستعلامات.. أما إذا أردت قصر التعامل مع الملفات من خلال استعلامات SQL فقط، فاختر الاختيار الثاني: Transact-SQL Access Enabled.. اضغط الزر Ok لإغلاق النافذة.. ستظهر لك رسالة تخبرك بأن بعض التغييرات لن تحدث إلا إذا أوقفت خادم سيكويل عن العمل وأعدت تشغيله Restart.. يمكنك فعل هذا من مدير تهيئة خادم سيكويل SQL Server Configuration Manager كما تعلمنا من قبل.

**٣- تفعيل استخدام مجرى البيانات FILESTREAM في قاعدة البيانات:**

يمكنك فعل هذا عند إنشاء قاعدة البيانات بجمل T-SQL وذلك باستخدام السمة FILESTREAM (هذا خارج نطاق هذا الكتاب).. كما يمكنك تعديل قاعدة البيانات بعد إنشائها لتفعيل هذه الخاصية، وذلك باستخدام الواجهة المرئية لمدير سيكويل SQL Server Management Studio.. افترض

أننا نتعامل مع قاعدة بيانات الكتب كمثال.. اتبع الخطوات التالية لتفعيل حفظ البيانات خارجها:

- افتح متصفح الكائنات Object Explorer في مدير سيكويل.
- ألحق Attach قاعدة البيانات C:\Books.mdf إن لم تكن موجودة.
- اضغط قاعدة بيانات الكتب بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط Properties.




- في نافذة الخصائص، اضغط العنصر FileGroups من القائمة اليسرى.. ستجد الجزء الأيمن مقسوما إلى نصفين:

أ. النصف العلوي يعرض مجموعات الملفات العادية (ملفات قاعدة البيانات الأساسية).. ويمكنك إضافة مجموعة أخرى لو أردت تقسيم قاعدة البيانات على أكثر من ملف.

ب. النصف السفلي يتعامل مع الملفات الخارجية Filestream.. هذا هو النصف الذي يعنينا.. اضغط الزر Add لإضافة صف جديد، وفي خانة الاسم اكتب FILESTREAM، وضع علامة الاختيار في الخانة Default كما هو موضح في الصورة.

- من القائمة اليسرى اضغط العنصر Files لعرض ملفات قاعدة البيانات، وفي الجهة اليمنى اضغط الزر Add لإضافة صف جديد.

- اكتب في خانة الاسم BooksFiles.. سيكون هذا هو اسم المجلد الذي سيتم حفظ الملفات التابعة لقاعدة البيانات فيه.
- في الخانة File Type أسدل قائمة العناصر واختر النوع FILESTREAM.. هذا سيجعل الخانة FileGroup تحتوي على القيمة FILESTREAM أيضا.
- استخدم الزر الموجود في الخانة Path لاختيار موضع حفظ المجلد.. هناك شرط إجباري عليك الالتزام به، وهو حتمية اختيار مسار على جزء من القرص الصلب مهيأ بتنسيق NTFS وليس FAT32.
- اضغط Ok لتنفيذ كل ما فعلناه.. ستجد أن مجلدا اسمه BooksFiles قد تم إنشاؤه على المسار الذي اخترته.. ويسمى هذا المجلد بحاوية الملفات Data Container، أو بمجموعة الملفات Filegroup.. لاحظ ما يلي:
  أ- لا يمكنك إنشاء مجموعات ملفات متداخلة.
  ب- لا يمكنك حفظ مجموعة الملفات على قسم مضغوط Compressed Volume من القرص الصلب.
  ج- لا يتم تشفير البيانات المحفوظة في الملفات الخارجية، حتى لو كانت قاعدة البيانات تقوم بتشفير بياناتها.. لهذا لو كان التشفير مهما بالنسبة لك، فأرسل البيانات مشفرة منذ البداية إلى قاعدة البيانات.
  د- يمكنك استخدام أوامر التحديث Update والحذف Delete والإدراج Insert الخاصة بلغة SQL للتعامل مع العمود الذي يحفظ بياناته في ملف خارجي، بنفس الطريقة التي تتعامل بها مع أي عمود عادي، حيث ستقوم قاعدة البيانات بإجراء هذه التغييرات على الملف الخارجي دون أن تشغل ذهنك بهذا.
  هـ- لا يمكنك استخدام الأمر UPDATE .Write لكتابة البيانات مجزأة في عمود يحفظ بياناته في ملف خارجي.. وبدلا من هذا عليك استخدام الفئة SqlFileStream التي سنتعرف عليها لاحقا، لكتابة أجزاء البيانات في الملف مباشرة.

**٤- تفعيل استخدام مجرى البيانات FILESTREAM في العمود:**

الآن يمكنك إنشاء عمود في جدول الناشرين (على سبيل المثال) يحفظ بياناته في ملفات خارجية.. لكن للاسف، لا يحتوي مصمم الجدول على خاصية تتيح لنا القيام بهذا بطريقة مرئية، لهذا ليس أمامنا سوى استخدام استعلام T-SQL.. لفعل هذا اتبع ما يلي:

- في متصفح الكائنات، اضغط اسم قاعدة البيانات BOOKS.MDF بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر New Query.
- في نافذة الاستعلامات، اكتب الاستعلام التالي:

```
ALTER TABLE Publishers
ADD Logo3 VARBINARY(MAX) FILESTREAM  NULL,
RowGuid UNIQUEIDENTIFIER
NOT NULL ROWGUIDCOL
UNIQUE DEFAULT NEWID()
GO
```

هذا الاستعلام يضيف إلى الجدول Publishers عمودين جديدين:

أ. العمود Logo3، وهو من النوع VARBINARY(MAX) وسيحفظ بياناته في ملف خارجي FILESTREAM، ويقبل القيمة NULL.

ب. والعمود RowGuid، وهو معرف متفرد للجدول UNIQUEIDENTIFIER، ولا يقبل العدم NOT NULL، وهو يعمل كمعرف لصفوف الجدول ROWGUIDCOL.. لاحظ أن وجود هذا العمود إجباري إذا أردت استخدام الملفات الخارجية، لأنه يستخدم في ربط الملف بالصف الذي يحفظ بياناته، لهذا لا يمكن أن يقبل هذا العمود القيمة Null.

ولتنفيذ هذا الاستعلام، اضغط في أي موضع من النافذة بزر الفأرة الأيمن، واضغط الأمر Execute.. سيظهر في الجزء السفلي من النافذة رسالة تخبرك بنجاح أو فشل التنفيذ.

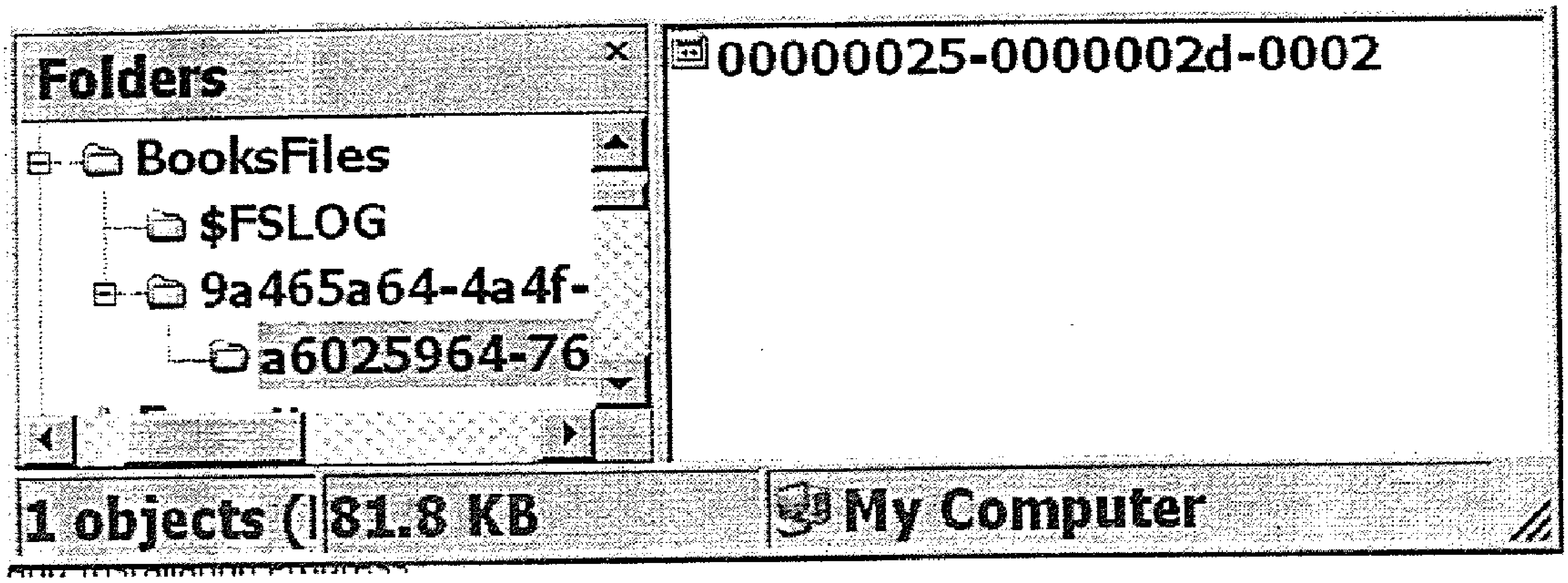
ويمكنك الاسترشاد بهذا الاستعلام في المواقف المماثلة، فكل ما عليك هو تغيير اسم الجدول Publishers واسم العمود Logo3 ليصير الاستعلام مناسبا لاحتياجاتك.

الآن فقط أنجزنا المهمة كاملة، وبإمكانك أن تحفظ صورة في الحقل Logo3 الخاص بالناشر الأول، باستعلام عادي كالتالي:

```
UPDATE Publishers
SET Logo3 =  @Logo
WHERE ID = 1
```

ويمكنك تنفيذ هذا الاستعلام بضغط الزر Write to FileStream في المشروع WriteLargeData، حيث سيظهر لك مربع حوار اختيار صورة.. اختر أي صورة تريدها واضغط OK.. الآن لو فتحت المجلد BooksFiles فستجد ملفا

جديدا قد أضيف إليه.. هذا الملف سيحمل اسما عجيبا لضمان عدم تشابه أسماء الملفات، لكنك لو أخذت نسخة منه وغيرت امتدادها إلى امتداد صورة (bmp. مثلا) فستجد أنها نفس الصورة التي اخترتها.




ولو عرضت بيانات جدول الناشرين، فستجد في الخانة التي حفظنا فيها الصورة أرقاما سداسية عشرية، تشير إلى موضع ملف الصورة في المجلد BooksFiles.
ويمكنك قراءة بيانات هذه الصورة باستعلام عادي كالتالي:

```
SELECT ID, Logo3
FROM Publishers
```

حيث يمكنك قراءة بيانات الصورة كاملة أو بطريقة تتابعية Sequential باستخدام قارئ البيانات DataReader كما سنرى لاحقا.. وهذا هو نفس ما يمكنك فعله مع البيانات الثنائية المحفوظة في قاعدة البيانات مثل image أو varbinary(MAX).. ويمكنك ضغط الزر Read FileStream في المشروع ReadLargeData لتجربة قراءة الصورة التى حفظتها خارج قاعدة البيانات، حيث سيتم حفظها على المحرك :\C بالاسم Logo1.bmp.

وهكذا نكون قد تعاملنا مع الملف الخارجي كأنه جزء من قاعدة البيانات.. يتبقى غذن أن نعرف كيف نتعامل مع هذا الملف مباشرة.. هذا هو دور الفئة SqlFileStream.. فلنتعرف عليها.

## فئة مجرى بيانات سيكويل SqlFileStream Class

هذه الفئة ترث الفئة Stream، مما يعني أنها تملك خصائص ووسائل القراءة من الملفات والكتابة فيها (راجع فصل الملفات في كتاب إطار العمل).. لكن هذه الفئة مخصصة للتعامل مع الملفات المحفوظة خارج قاعدة البيانات من خلال السمة FILESTREAM.

ولحدث إنشاء هذه الفئة صيغتان:

١- الصيغة الأولى تستقبل ثلاثة معاملات، هي بالترتيب:

- نص يمثل مسار الملف.. ويمكنك الحصول على مسار الملف من خلال استعلام SQL، باستخدام الوسيلة ( )PathName. التي تستخدم مع اسم العمود الذي يحفظ بياناته في ملفات خارجية.
- مصفوفة وحدات ثنائية Byte Array تحتوي على محتوى التعامل Transaction Context.. ويمكنك الحصول على مسار الملف من خلال استعلام SQL، باستخدام الوسيلة GET_FILESTREAM_TRANSACTION_CONTEXT..

  لاحظ أنك لا تستطيع إرسال القيمة Nothing إلى هذا المعامل.. هذا معناه أن عليك التعامل مع الملف من خلال تعامل Transaction.. هذا يلفت انتباهك إلى أن سيكويل سيرفر ما زال ينظم عملية الكتابة في الملف، وإن حدث أي خطأ فسيلغي كل التغييرات، ولن يتم حفظ هذه التغييرات إلا إذا قمت بإجراء التعاملات Commit Transactions، كما سنعرف لاحقا
- إحدى قيم المرقم FileAccess التي توضح الطريقة التي تريد التعامل بها مع الملف: للقراءة Read، للكتابة Write، أم للقراءة والكتابة معا ReadWrite.

٢- الصيغة الثانية تزيد على الصيغة الأولى بمعاملين إضافيين:

- معامل من نوع المرقم FileOptions، يحدد خيارات التمعامل مع الملف.. وقد تعرفنا على هذا المرقم في فصل الملفات في كتاب برمجة إطار العمل، لكن ما يهمنا من قيمه هنا هو القيمة SequentialScan، فهي تتيح لنا قراءة أجزاء من الملف على التوالي، والقيمة RandomAccess التي تتيح لنا القراءة من أي موضع في الملف دون ترتيب.

- عدد صحيح يستقبل الحجم المبدئي الذي تريد حجزه للملف عند إنشائه.. وإذا أردت استخدام القيمة الافتراضية الخاصة بدوت نت، فأرسال إلى هذا المعامل القيمة صفر.

وإضافة إلى ما ترثه من الفئة الأم من وسائل وخصائص، تمتلك هذه الفئة الخاصيتين التاليتين:

**الاسم Name:**

تعيد مسار الملف المرسل إلى المعامل الأول في حدث الإنشاء.

**محتوى التعامل TransactionContext:**

تعيد مصفوفة محتوى التعامل المرسلة إلى المعامل الثاني في حدث الإنشاء.

وستجد مثالا على استخدام هذه الفئة في الزر SqlFileStream.Read في المشروع ReadLargeData.. في هذا الزر نستخدم الاستعلام التالي:

```
SELECT ID, Logo3.PathName( ),
GET_FILESTREAM_TRANSACTION_CONTEXT ( )
From Publishers
```

هذا الاستعلام يعيد ثلاثة حقول، هي بالترتيب: رقم الناشر، مسار ملف الصورة، مصفوفة محتوى التعامل Transaction Context الخاصة بالملف.. ويستخدم الكود هذه الحقول لفتح الملف وقراءة محتوياته، وحفظها في ملف جديد على المسار :C.

وستجد مثالا على استخدام هذه الفئة لحفظ صورة في الخانة Logo3 الخاصة بالناشر الثاني، وذلك في الزر SqlFileStream.Write في المشروع WriteLargeData.. في هذا الزر استخدمنا استعلاما شبيها بالاستعلام السابق، لكننا لجأنا أولا إلى حيلة صغيرة، فقد استخدمنا استعلاما آخر لوضع القيمة Null في الملف، وفصلنا الاستعلامين بالفاصلة المنقوطة:

```
UPDATE Publishers
SET Logo3 = 0x0
Where ID = 2;
SELECT Logo3.PathName( ),
GET_FILESTREAM_TRANSACTION_CONTEXT ( )
From Publishers
Where ID = 2
```

الحكمة من وراء هذا، هو أن خانة الصورة لو كانت فارغة فلن يكون هناك ملف مرتبط بها، وستعيد الوسيلة PathName القيمة Null وبالتالي لن نستطيع الكتابة في الملف بالفئة SqlFileStream.. لهذا سنضع في الخانة القيمة 0x0.. هذا يجعلها تنشئ ملفا وتتركه فارغا، لكن ما يعنينا هنا هو أننا نستطيع معرفة مساره للكتابة فيه.. أما لو كان هناك ملف فعلا وبه بيانات، فسيتم محوها، وهذا يناسبنا فعلا، لأننا سنكتب بيانات جديدة.

أما لو أردت إضافة بيانات إلى نهاية ملف موجود، فستحتاج أولا إلى استخدام الوسيلة SqlFileStream.Seek للوصول إلى نهاية هذا الملف قبل بدء الكتابة فيه، بنفس الطريقة التقليدية التي تتعامل بها مع الملفات العادية:

**SqlFileStream.Seek(0, SeekOrigin.End)**

طبعا في هذه الحالة لن تستخدم الاستعلام الذي يضع القيمة 0x0 في الخانة، لأنك تريد المحافظة على بيانات الملف الموجود.

# كـائـــــن الاتــصـــــال

# Connection Object

يتيح لك هذه الكائن الاتصال بقاعدة بيانات تعمل على الخادم.. وكما ذكرنا سابقا، سنركز هنا على دراسة الفئة sqlConnection، لهذا لا تـنسَ إضـافة جملـة التضمين التالية أعلى صفحة الكود:

**Imports System.Data.SqlClient**

## نص الاتصال Connection String:

للاتصال بسيكويل سيرفر، يجب أن ترسل إليه نصا يحتوى على البيانات اللازمة، مثل اسم قاعدة البيانات، واسم المستخدم وكلمة السر.. ويتكون نص الاتصال من مجموعة من القيم، يفصل بين كل منها العلامة ; وذلك على الصيغة:

Property1 = Value1;  Property2 = Value2; .............
PropertyN = ValueN

على سبيل المثال، النص التالي هو نص الاتصال بقاعدة بيانـات الكتـب علـى سيكويل سيرفر:

```
Data Source = .\SQLEXPRESS;
AttachDbFilename = C:\Books.mdf;
Integrated Security = True;
Connect Timeout = 30;
```

| ملحوظة: |
| --- |
| عند بناء نص الاتصال بمزود ODBC، عليك وضع القيم بين قوسين متعرجين { }، على الصيغة:<br>Property1 = {Value1}; Property2 = {Value2}; ....... PropertyN = {ValueN}<br>بينما في باقي المزودات يمكنك استخدام علامات التنصيص بدلا من الأقواس المتعرجة. |

وتختلف بعض الخصائص المرسلة عبر نص الاتصال، تبعا لنوع مزود قاعدة البيانات المستخدم.. وقد كانت كتابة نص الاتصال تمثل مشكلة قبل ظهور الإصدار الثاني من إطار العمل مع دوت نت ٢٠٠٥، حيث وفر إطار العمل فئة خاصة تسمى "باني نص الاتصال" DbConnectionStringBuilder، ومنها تم اشتقاق فئة لبناء نص اتصال كل مزود من مزودات قاعدة البيانات.. وهذه الفئات هي:

| الفئة | وظيفتها |
| --- | --- |
| SqlConnectionStringBuilder | باني نص اتصال سيكويل سيرفر. |
| OleDbConnectionStringBuilder | باني نص اتصال OLEDB. |
| OdbcConnectionStringBuilder | باني نص اتصال ODBC. |
| OracleConnectionStringBuilder | باني نص اتصال أوراكل. |

## فئة باني نص الاتصال
## DbConnectionStringBuilder Class

هذه الفئة موجودة في النطاق System.Data.Common، وهي تمثل واجهة القاموس IDictionary، مما يعني أنها مجموعة Collection كل عنصر من عناصرها يكون في صورة مفتاح Key وقيمة Value.. وبهذا التصميم تستطيع تكوين نص الاتصال، بإضافة عناصر إلى هذا القاموس، كل عنصر منها يتكون من اسم خاصية الاتصال وقيمتها، حيث ستقوم الفئة DbConnectionStringBuilder بتكوين صيغة نص الاتصال بناء على هذه العناصر.

وتوجد هذه الفئة في نطاق الاسم System.Data.Common، لهذا لا تنسَ إضافة الجملة التالية أعلى صفحة الكود قبل استخدامها:

**Imports System.Data.Common**

ولحدث الإنشاء Constructor الخاص بهذه الفئة صيغتان:

١- الأولى لا تستقبل أي معاملات.. مثال:

**Dim CsB As New DbConnectionStringBuilder**

٢- والثانية تستقبل معاملا منطقيا Boolean، إذا جعلت قيمته True فسيتم وضع القيم المكتوبة في نص الاتصال بين قوسين مضلعين { } لاستخدامه مع مزود ODBC.

وبالإضافة إلى خصائص القاموس الشهيرة، تمتلك هذه الفئة الخاصيتين التاليتين:

### نص الاتصال ConnectionString:

تقرأ أو تغير نص الاتصال الذي تتعامل معه هذه الفئة.. وتستطيع الحصول على نص الاتصال أيضا باستخدام الوسيلة ToString.. لاحظ أن باني نص الاتصال يرتب المفاتيح في النص العائد حسب أولويتها، وليس على حسب ترتيب إضافتك لها.

### نص اتصال قابل للتصفح BrowsableConnectionString:

إذا جعلت قيمة هذه الخاصية True، فسيتم عرض نص الاتصال في نافذة الخصائص عندما تستخدم الأداة PropertyGrid لعرض خصائص باني نص الاتصال.

وبالإضافة إلى وسائل القاموس الشهيرة، تمتلك هذه الفئة الوسائل التالية:

### إضافة مفتاح وقيمة AppendKeyValuePair:

تتيح لك إضافة خاصية وقيمتها إلى نص اتصال موجود في باني نص StringBuilder، حيث ستقوم بتكوين الصيغة الصحيحة للخاصية والقيمة، ثم إضافتها في نهاية باني النص.

وتستقبل هذه الوسيلة ثلاثة معاملات: باني النص StringBuilder، ونصا يمثل اسم الخاصية، ونصا يمثل قيمتها.. مثال:

```
Dim SB As New System.Text.StringBuilder(
        "Data Source = .\SQLEXPRESS;")
DbConnectionStringBuilder.AppendKeyValuePair(SB,
        "AttachDbFilename", "C:\Books.mdf")
DbConnectionStringBuilder.AppendKeyValuePair(SB,
        "Integrated Security", "True")
MsgBox(SB.ToString)
```

ستعرض الرسالة النص:

Data Source = .\SQLEXPRESS; AttachDbFilename = C:\Books.mdf; Integrated Security=True

وتوجد صيغة أخرى لهذه الوسيلة، تزيد على الصيغة الأولى بمعامل رابع، إذا جعلت قيمته True، فسيتم وضع القيم بين قوسين متعرجين {} لاستخدام نص الاتصال مع مزود ODBC.

### مساوٍ لـ EquivalentTo:

أرسل إلى هذه الوسيلة نسخة من الفئة DbConnectionStringBuilder لمقارنتها بالنسخة الحالية من الفئة DbConnectionStringBuilder.. وتتم المقارنة بالتأكد من أن كل مفتاح في القاموس الأول له ما يناظره في القاموس الثاني (بغض النظر عن الترتيب)، وأن القيمتين المحفوظتين في كليهما متساويتان.. لاحظ أن مقارنة المفاتيح لا تراعي حالة الأحرف، بينما مقارنة القيم تراعي حالة الأحرف.. وفي حالة نجاح المقارنة يعتبر نصا الاتصال الموجودين في القاموسين متساويين، وتعيد هذه الوسيلة True.. وستجد مثالا على هذه الوسيلة في الزر EquivalentTo في المشروع ConStrBuilder.

- **هل يحتوي على ShouldSerialize:**

تعيد True إذا كان المفتاح الذي أرسلته إليها كمعامل موجودا ضمن نـص الاتصـال.. هذا معناه أن هذه الوسيلة مكافئة تماما للوسيلة ContainsKey.. وستجد مثالا على هذه الوسيلة في الزر ShouldSerialize في المـشروع ConStrBuilder.

- **محاولة معرفة القيمة TryGetValue:**

تحاول قراءة قيمة أحد المفاتيح الموجودة في نص الاتصـال، فإن كان المفتاح موجودا أعادت True، وإن لم يكن موجودا أعادت False دون أن تـسبب خطأ في البرنامج.. لهذا يعتبر استخدامها أفضل من استخدام الخاصية Item لقراءة قيمة المفتاح، فهي تسبب خطأ إن لم يكن المفتاح موجودا، مما يستلزم استخدام الوسيلة ContainsKey أولا على سبيل الاحتياط.. مثلا:

```
If CSB.ContainsKey("AttachDbFilename") Then
  MsgBox(CSB("AttachDbFilename")) ' C:\Books.mdf
End If
```

وللوسيلة TryGetValue معاملان: الأول معامل نصـي يستقبل اسم المفتاح، والثاني معامل مرجعي ByRef من النوع Object، يعيد إليك قيمة المفتاح إن وجد.. والكود التالي هو إعادة كتابة للمثال السابق باستخدام هذه الوسيلة:

```
Dim Value As Object
If CSB.TryGetValue("AttachDbFilename", Value) Then
  MsgBox(CSB("AttachDbFilename")) ' C:\Books.mdf
End If
```

## فئة باني نص اتصال سيكيول
## SqlConnectionStringBuilder Class

هذه الفئة ترث الفئة DbConnectionStringBuilder، ويمكنك استخدامها لبناء نص الاتصال بسيكويل سيرفر، فهي تمتلك المزيد من الخصائص التي تحمل أسماء المعلومات اللازمة للاتصال بسيكويل سيرفر، مما يجعل تكوين نص الاتصال في منتهى السهولة والوضوح.

ولحدث إنشاء هذه الفئة صيغتان:

١- الأولى بدون معاملات.

٢- والثانية تستقبل نص اتصال لإضافته إليها مبدئيا، حيث يمكنك إضافة أي تفاصيل أخرى إليه بعد ذلك.

وبجوار ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

### توصيل ملف قاعدة البيانات AttachDBFilename:

تناظر المفتاح AttachDBFilename أو initial file name في نص الاتصال.. ويمكنك أن تضع في هذه الخاصية مسار واسم الملف الأساسي لقاعدة البيانات التي تريد الاتصال بها، وفي هذه الحالة سيتم توصيل هذه القاعدة بالخادم، والاتصال بها.

وعليك أن تتأكد أن ملف قاعدة البيانات ليس للقراءة فقط Read Only، لأن توصيل قاعدة البيانات يحتاج إلى إنشاء ملف سجل الأداة Log، واسمه يوضع في ملف قاعدة البيانات، ولو كانت للقراءة فقط فسيحدث خطأ ولن ينجح الاتصال.

أيضا، قد يحدث خطأ إذا كان ملف سجل الأداء Log موجودا في مجلد قاعدة البيانات وأنت تحاول توصيلها، مع وجود المفتاح Database في نص الاتصال.. في هذه الحالة عليك حذف ملف سجل الأداء وإعادة الاتصال، حيث سيتم إنشاء سجل أداء جديد لقاعدة البيانات.

### الفهرس الأساسي InitialCatalog:

تقرأ أو تغير اسم قاعدة البيانات التي تريد الاتصال بها على الخادم.. وتختلف هذه الخاصية عن الخاصية السابقة في أنها تتعامل مع قاعدة بيانات متصلة بالخادم فعلا في هذه اللحظة، لهذا يتم ذكر اسم قاعدة البيانات فقط بدون المسار والامتداد (مثل Books).. بينما في الخاصية

AttachDBFilename يتم ذكر مسار ملف قاعدة البيانات لتوصيلها بالخادم ثم الاتصال بها.
وتناظر هذه الخاصية المفتاح database أو Initial Catalog في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها نصا فارغا.

**مصدر البيانات DataSource:**
تقرأ أو تغير عنوان خادم سيكويل.. قد يكون هذا العنوان للخادم المحلي .\SQLEXPRESS، أو لخادم بعيد Remote Server له عنوان بروتوكول الإنترنت IP Address الخاص به مثل (10.0.0.127).
وتناظر هذه الخاصية المفتاح Data Source أو server أو address أو addr أو network address في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها نصا فارغا.

**بديل فشل الاتصال FailoverPartner:**
تقرأ أو تغير عنوان خادم سيكويل البديل، الذي سيتم استخدامه إذا فشل الاتصال بالخادم الرئيسي الموضح في الخاصية السابقة.
وتناظر هذه الخاصية المفتاح Failover Partner في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها نصا فارغا.

**حماية متكاملة IntegratedSecurity:**
تناظر المفتاح trusted_connection أو Integrated Security في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها False، مما يعني أن عليك إمداد الاتصال باسم المستخدم وكلمة المرور.. أما لو جعلت قيمتها True، فسيتم استخدام حساب المستخدم على الويندوز للاتصال.. هذا مفيد عند الاتصال بالخادم المحلي، أو عند استخدام البرنامج داخل شركة تستخدم شبكة داخلية LAN، ففي هذه الحالة يقوم مدير نظام سيكويل سيرفر System Administrator بتعريف حسابات الويندوز الخاصة بأجهزة المستخدمين المتصلة بالشبكة والمسموح لها بالاتصال بالخادم، وبهذا يكفي مجرد تسجيل الدخول على الويندوز لضمان سرية الاتصال بالخادم.

**معرف المستخدم UserID:**
تقرأ أو تغير اسم المستخدم الذي يتصل بالخادم، وذلك في حالة عدم استخدام الحماية المتكاملة Integrated Security.

وتناظر هـذه الخاصـية المفتـاح User ID أو user أو uid فـي نـص الاتصـال.. وفي الوضع الافتراضي تكون قيمتها نصا فارغا.

**كلمة السر Password:**

تقرأ أو تغير كلمة المرور اللازمة للاتصال بالخادم، وذلك في حالـة عـدم استخدام الحماية المتكاملة Integrated Security.

وتناظر هذه الخاصية المفتاح Password أو pwd في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها نصا فارغا.

**معرف الجهاز WorkstationID:**

تقرأ أو تغير اسم الجهاز الذي يتـصل بالخـادم، وهـي تتـاظر المفتـاح Workstation ID أو wsid في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها نصا فارغا.

**إبقاء معلومات السرية PersistSecurityInfo:**

تناظر المفتـاح Persist Security Info أو persistsecurityinfo فـي نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها False، مما يعني أن عليك إرسال اسم المستخدم وكلمة السر كلما أردت فتح الاتصال.. أمـا لـو جعلت قيمتها True، فيمكنك إرسال هذه المعلومات عند فتح الاتصال لأول مرة فقط، وسيتم الاحتفاظ بها لاستخدامها في فتح الاتصال بعد هذا.

**نفاد وقت الاتصال ConnectTimeout:**

تمثل وقت الانتظار الذي ستعتبر محاولة الاتصال بالخادم فاشلة بعد مروره دون استجابة من الخادم، وهي تتـاظر المفتـاح Connect Timeout أو connection timeout أو timeout في نص الاتصال.. وفـي الوضـع الافتراضي تكون قيمتها ١٥ ثانية.

**اسم التطبيق ApplicationName:**

تناظر المفتاح app أو Application Name في نص الاتـصال.. وفـي الوضع الافتراضي تكون قيمتها .NET SqlClient Data Provider، لكن بإمكانك أن تضع فيها اسم برنامجك.

## اللغة الحالية CurrentLanguage:

تقرأ أو تغير اسم سجل اللغة في سيكويل سيرفر، وهي تناظر المفتاح language أو Current Language في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها نصا فارغا "".

## التشفير Encrypt:

تناظر المفتاح Encrypt في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها False مما يعني أن خادم سيكويل لن يشفر البيانات المرسلة بينه وبين العميل.. ولو جعلت قيمتها True، فسيتم استخدام نوع من التشفير يسمى SSL Encryption الذي يعني "تشفير طبقة مقابس الاتصال الآمنة" Secure Sockets Layer Encryption، وهو يستخدم مفتاحين: مفتاحا عاما Public Key يستخدم لتشفير البيانات، ومفتاحا خاصا Private Key يستخدم لفك تشفيرها.

## إجازة خادم موثوق به TrustServerCertificate:

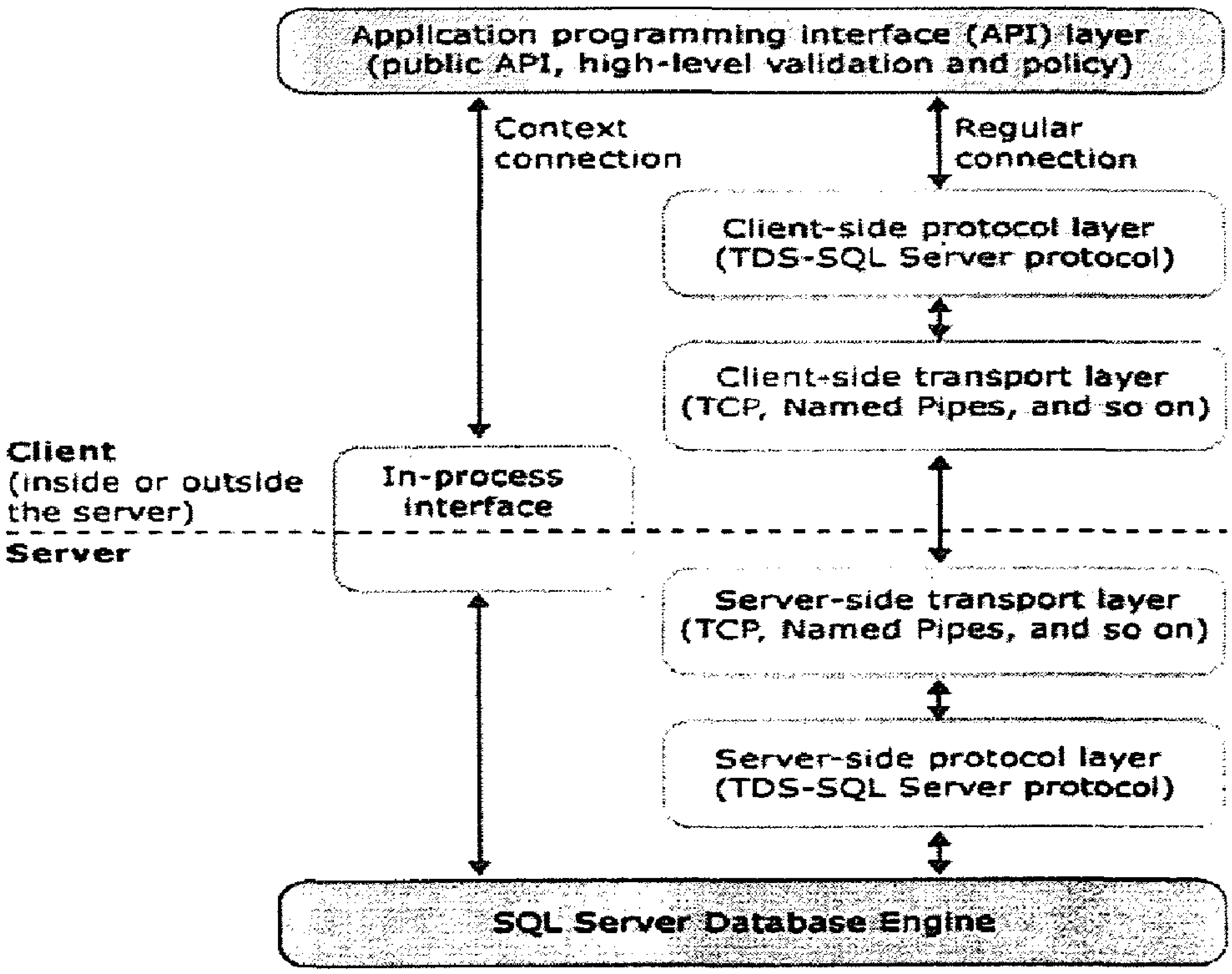
تناظر المفتاح TrustServerCertificate في نص الاتصال، وإذا جعلت قيمتها True، فسيتم تجاهل عملية إجازة الخادم Certification، اكتفاءً بحماية البيانات باستخدام تشفير SSL.

## اتصال بالمحتوى ContextConnection:

تناظر المفتاح Context Connection في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها False.

وينصح بجعل قيمة هذه الخاصية True عند الاتصال بخادم محلي Local Server توجد عليه الإجراءات المخزنة ودوال T-SQL التي تريد تنفيذها، لأن هذا يجعلك تستخدم نفس موارد الاتصال السابق بالخادم المحلي، مما يوفر عليك إعادة إدخال اسم المستخدم وكلمة السر، ويتيح لك التفاعل مع التعاملات Transactions التي لم يتم حفظها بعد، كما يتيح لك استخدام الجداول المؤقتة التي تم إنشاؤها على الاتصال المحلي.. وتؤدي هذه الطريقة إلى أداء أفضل للبرنامج، لأنها تتجاهل بروتوكولات الشبكة ومراحل نقل البيانات عبرها، وتتعامل مباشرة مع الخادم المحلي (لأنه يوجد على نفس الجهاز)، مما يجعل الاتصال أسرع وأكفأ.

وينصح بجعل قيمة هذه الخاصية False لاستخدام الاتصال العادي Regular Connection، وذلك عند الاتصال بخادم بعيد (غير محلي) Remote Server.
والشكل التالي يلخص الفارق بين هذين النوعين من الاتصال.. لاحظ أن الاتصال بالمحتوى يتجاهل العديد من طبقات الاتصال عبر الشبكة Network Layers، ويستخدم واجهة الاتصال المباشر In-Process Interface.




### في القائمة Enlist:

تناظر المفتاح Enlist في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها True، مما يعني أن الاتصال الحالي سيوضع في قائمة الاتصالات المستخدمة لمحتوى التعاملات الحالي Current Transaction Context..
هذا يسمح بجعل أكثر من اتصال تتشارك في تعامل Transaction واحد، بحيث لو فشلت أي عملية على أي اتصال منها يتم إلغاء العمليات التي تمت على باقي الاتصالات.. لن نتعمق في موضوع التعاملات Transactions في هذا الكتاب، وسنتعرف عليه مع باقي المواضيع المتقدمة في برمجة قواعد البيانات في الكتاب القادم بإذن الله.

### معالجة غير متزامنة AsynchronousProcessing:

تناظر المفتاح async أو Asynchronous Processing في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها False، ولو جعلت قيمتها True فسيعني هذا السماح للخادم بإجراء عمليات معالجة غير متزامنة.. هذا معناه أن برنامجك سيواصل العمل مباشرة بعد إرسال الاستعلام إلى الخادم، تاركا الخادم يواصل تنفيذ الاستعلام.. هذا يوفر عليك كتابة الكثير من الكود لإنشاء عمليات فرعية Threads أو عمليات غير متزامنة في برنامجك لضمان مواصلة الاستجابة للمستخدم أثناء معالجة سيكويل سيرفر للاستعلامات السابقة.

### مساهمة Pooling:

تناظر المفتاح Pooling في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها True، مما يعني أن هذا الاتصال سينضم إلى رصيد الاتصالات المساهمة Connection Pool التي تظل مفتوحة دائما لاستخدامه فور الحاجة إليها.. أما لو جعلت قيمتها False فسيعني هذا أن هذا الاتصال سيتم فتحه وإغلاقه مباشرة بعد انتهاء استخدامه، وعند الاحتياج إليه مجددا يتم فتحه من جديد.. وهكذا.

### أقصى حجم للمساهمة MaxPoolSize:

تناظر المفتاح Max Pool Size في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها ١٠٠، مما يعني الاحتفاظ في رصيد الاتصالات المساهمة المتاحة للاستخدام، بمئة اتصال ــ كحد أقصى ــ مفتوحة بين الخادم والعميل.

### أقل حجم للمساهمة MinPoolSize:

تقرأ أو تغير أصغر عدد من الاتصالات يجب أن يظل مفتوحا بين الخادم والعميل في رصيد الاتصالات المساهمة، وهي تناظر المفتاح Min Pool Size في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها صفرا.

### وقت انتظار توازن الحمل LoadBalanceTimeout:

تقرأ أو تغير الوقت بالثانية، الذي يتم انتظاره أثناء وجود الاتصال في رصيد الاتصالات المساهمة Connection Pool، قبل أن يتم إغلاق الاتصال،

وذلك لضمان عدم ترك الاتصالات المفتوحة خاملة بدون استخدام لفترات طويلة.. وفي الوضع الافتراضي تكون قيمتها صفرا، مما يعني ترك الاتصال مفتوحا دائما بدون قيود.

وتناظر هذه الخاصية المفتاح connection lifetime أو Load Balance Timeout في نص الاتصال.

### إعادة الاتصال إلى وضعه الأصلي ConnectionReset:

تناظر المفتاح Connection Reset في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها True، وهذا معناه أن الاتصال سيعود إلى وضعه الأصلي عند طلب استخدامه من رصيد الاتصالات المساهمة Connection Pool.

### مجموعات النتائج الفعالة المتعددة MultipleActiveResultSets :

تناظر المفتاح MultipleActiveResultSets في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها False، وهذا معناه استخدام "مجموعة النتائج العادية Default Result Set"، وفيها يتم إرسال نتائج الاستعلام من خادم سيكويل إلى جهاز العميل، حيث يتم حفظها في مخزن وسيط Buffer في الذاكرة، وعندما يحتاج برنامجك إلى عرضها للمستخدم، يتم المرور عبرها سجلا بسجل.. ولا يستطيع العميل استخدام الاتصال المفتوح مع الخادم في تحديث البيانات قبل أن ينتهي من التعامل مع كل البيانات التي أرسلها الخادم أولا، أو قبل أن يرسل إلى الخادم طلبا لإلغاء إرسال باقي النتائج.. وتعتبر هذه الطريقة أكثر كفاءة في استغلال الاتصال، لأن الخادم يرسل أكبر كم ممكن من النتائج عبر حزم البيانات Packets المرسلة عبر الشبكة Network.

ولو جعلت قيمة هذه الخاصية True، فسيتم استخدام "مجموعات النتائج الفعالة المتعددة Multiple Active Result Sets" أو اختصارا MARS، وهي متاحة فقط مع سيكويل سيرفر ٢٠٠٥ وما يليه من إصدارات، وفيها يسمح للعميل باستخدام أكثر من قارئ بيانات SqlDataReader في نفس الوقت.

### مكتبة الشبكة NetworkLibrary:

ضع في هذه الخاصية اسم مكتبة الربط DLL التي تريد من الخادم استخدامها للاتصال عبر الشبكة، وذلك بشرط توفر هذه المكتبة على الخادم.

وتناظر هذه الخاصية المفتاح Network Library أو network أو net في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها نصا فارغا.. والجدول التالي يوضح القيم المحتملة لهذه الخاصية:

| اسم مكتبة الربط | نوع الاتصال |
|---|---|
| dbnmpntw | Named Pipes |
| dbmsrpcn | Multiprotocol |
| dbmsadsn | AppleTalk |
| dbmsgnet | VIA |
| dbmslpcn | Shared Memory |
| dbmsspxn | IPX/SPX |
| dbmssocn | TCP/IP |

وفي حالة التعامل مع خادم محلي وترك قيمة هذه الخاصية فارغة، يتم استخدام المكتبة dbmslpcn (Shared Memory).

### حجم حزمة البيانات PacketSize:

عند إرسال البيانات عبر الشبكة أو الإنترنت، يتم تقسيمها إلى حزم Packets.. وتحدد هذه الخاصية حجم كل حزمة من هذه الحزم بالوحدة الثنائية Byte، وفي الوضع الافتراضي تكون قيمتها ٨٠٠٠ وحدة Byte. وتناظر هذه الخاصية المفتاح Packet Size في نص الاتصال.

### النسخ المطابق Replication:

تناظر المفتاح Replication في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها False، وإذا جعلتها True فسيتم تمكين عملية النسخ المطابق Replication عبر هذا الاتصال، وهي تقنية آلية تتيح لك نسخ قاعدة بيانات بين أكثر من خادم، وإبقاء البيانات متزامنة بين الخادمين، بحيث يتم تحديث إحدى قاعدتي البيانات إذا حدث تغيير في الأخرى.. هذا مفيد عندما يكون هناك خادم أصلي وخادم احتياطي للتعامل معه إذا حدثت مشكلة في الخادم الأصلي أو تم إيقافه للصيانة مثلا.

### ربط التعاملات TransactionBinding:

تناظر المفتاح Transaction Binding في نص الاتصال.. ويمكنك أن تضع فيها إحدى القيمتين التاليتين:

| | |
|---|---|
| Implicit Unbind | فك ارتباط ضمني: وهي القيمة الافتراضية، وفيها يؤدي إغلاق الاتصال إلى فصله عن التعاملات الجارية Current Transactions. |
| Explicit Unbind | فك ارتباط صريح: يجب عليك فك الارتباط بين الاتصال والتعاملات الجارية بطريقة صريحة قبل إغلاق الاتصال، وإلا حدث خطأ. |

### إصدار نظام الأنواع TypeSystemVersion:

تناظر المفتاح Type System Version في نص الاتصال، وهي تتيح لك تحديد إصدار سيكويل سيرفر الذي تريد أن تستخدم أنواع البيانات الخاصة به.. على سبيل المثال، لو كان الخادم يستخدم سيكويل سيرفر ٢٠٠٨، وجعلت قيمة هذه الخاصيةSQL Server 2000، فيمكنك استخدام أنواع البيانات Data Types الخاصة بسيكويل سيرفر ٢٠٠٠، حيث سيقوم سيكويل سيرفر ٢٠٠٨ بإجراء التحويلات المناسبة للتعامل معها.. والجدول التالي يلخص لك القيم الممكنة لهذه الخاصية:

| | |
|---|---|
| SQL Server 2000 | يتم استخدام الأنواع الخاصة بسيكويل سيرفر ٢٠٠٠.. هذا يعني إجراء بعض التحويلات عند التعامل مع إصدارات أحدث، مثل:<br>- تحويل XML إلى NTEXT.<br>- تحويل UDT إلى VARBINARY.<br>- تحويل VARCHAR(MAX) إلى TEXT.<br>- تحويل NVARCHAR(MAX) إلى NEXT.<br>- تحويل ARBINARY(MAX) إلى IMAGE. |
| SQL Server 2005 | يتم استخدام الأنواع الخاصة بسيكويل سيرفر ٢٠٠٥. |
| SQL Server 2008 | يتم استخدام الأنواع الخاصة بسيكويل سيرفر ٢٠٠٨. |
| Latest | يتم استخدام أحدث إصدار من سيكويل سيرفر يمكن للخادم والعميل التعامل معه. |

### نسخة المستخدم UserInstance:

تناظر المفتاح User Instance في نص الاتصال.. وفي الوضع الافتراضي تكون قيمتها False، ولو جعلتها True فسيتم توجيه الاتصال من نسخة خادم سيكويل الافتراضية، إلى نسخة أخرى مخصصة للعميل، لكن هذا قد يسبب أخطاء في الاتصال إذا كنت تستخدم السمة FILESTREAM لحفظ بيانات بعض الأعمدة في ملفات خارجية.

ولا تمتلك هذه الفئة أية وسائل Methods غير ما ترثه من الفئة الأم.

والمثال التالي يريك كيف تستخدم هذه الفئة لتكوين نص اتصال بقاعدة الكتب على الخادم المحلي باستخدام الحماية المتكاملة:

```
Dim CnStrBldr As New SqlConnectionStringBuilder
CnStrBldr.DataSource = ".\SQLEXPRESS"
CnStrBldr.InitialCatalog = "Books"
CnStrBldr.IntegratedSecurity = True
Dim CnStr = CnStrBldr.ConnectionString
MsgBox(CnStr)
```

## حفظ نص الاتصال في إعدادات البرنامج Settings:

عند كتابة برنامج يتعامل مع قواعد البرنامج، يلجأ المبرمج في معظم الأحول إلى إنشاء قاعدة البيانات على جهازه، أو ينسخ جزءا من قاعدة البيانات من الخادم إلى جهازه، ليجعلها تعمل على الخادم المحلي، ومن ثم يتصل بها من برنامجه.. هذا يجعل كتابة واختبار الكود وتصحيحه أسرع من التعامل مع خادم حقيقي عبر شبكة الإنترنت، كما أنه يضمن عدم تخريب قاعدة البيانات الرئيسية عند إضافة أو حذف السجلات للاختبار.

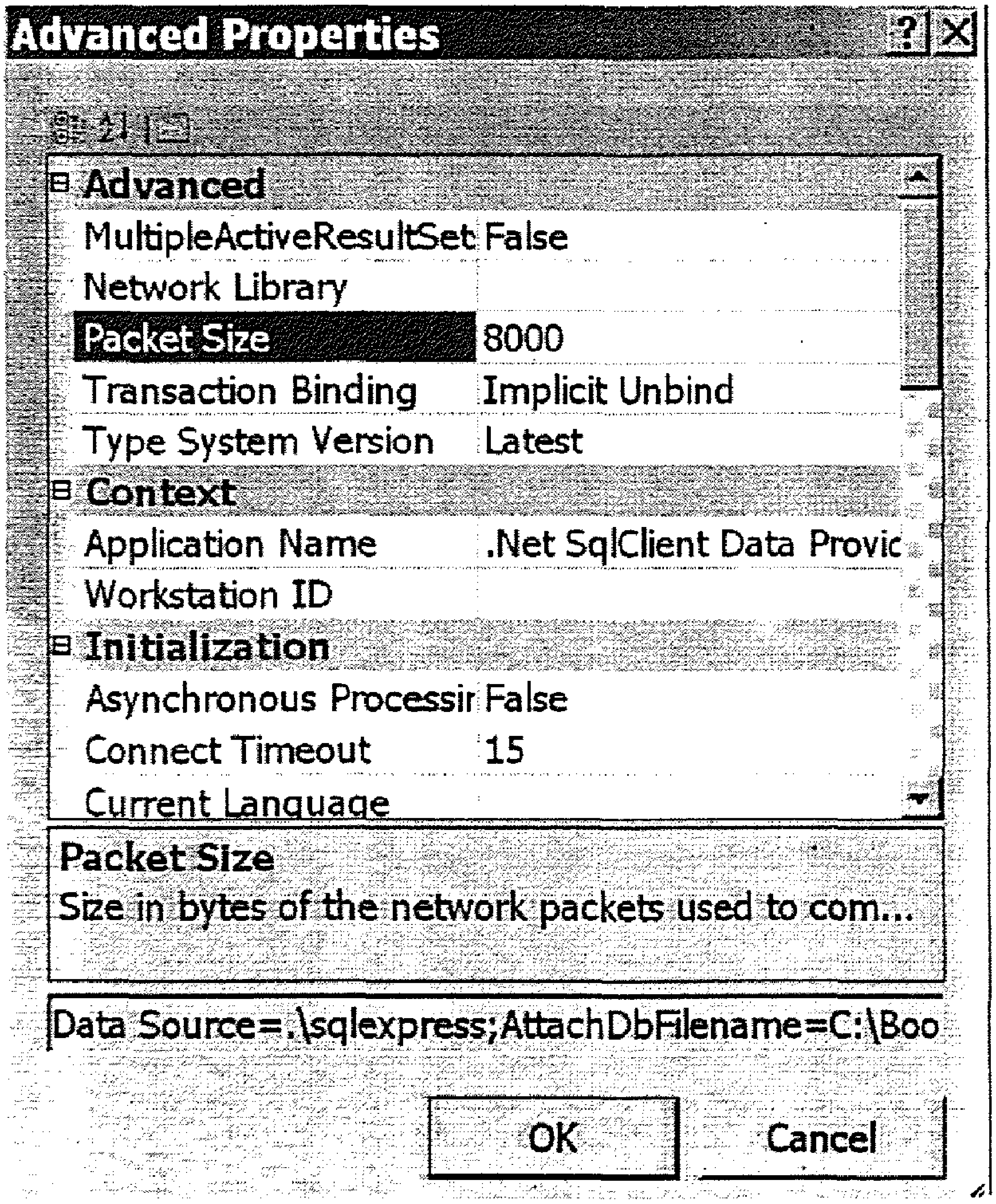
وبعد الانتهاء من البرنامج، يتم رفع قاعدة البيانات إلى الخادم (إن لم تكن موجودة عليه)، وتصحيح نصوص الاتصال لتشير إلى الخادم الحقيقي بدلا من الخادم المحلي، لكي يبدأ البرنامج عمله في صورته النهائية.

ونظرا لأن البرامج العملية قد تتعامل مع أكثر من قاعدة بيانات، كما أن عنوان قاعدة البيانات قد يتغير في أي لحظة لو تم نقلها من خادم إلى آخر على الإنترنت، يصير من غير العملي كتابة نص الاتصال في الكود، لأن البحث عنه وتغييره في كل المواضع أمر مرهق وعرضة للخطأ.. لهذا يفضل كتابة نصوص الاتصال في ملف خارج البرنامج، مع استخدام إحدى طرق التشفير لحماية التفاصيل المكتوبة به (كاسم المستخدم وكلمة السر).. ويمكن فعل هذا يدويا، أو باستخدام إحدى الطرق الجاهزة التي تمنحها دوت نت، كالإعدادات Settings التي تعرفنا عليها بالتفصيل الممل في مرجع "برمجة نماذج الويندوز"، وقلنا هناك إن دوت نت تقدم رعاية خاصة لنصوص الاتصال، فقد خصصت لها مقطعا خاصا في ملف الإعدادات، ومنحتنا فئة تتعامل معه، هي الفئة ConnectionStringsSection.. ولقد أرجأنا شرح هذا الموضوع إلى حين التعرف على قواعد، وها نحن أولاء ☺ .

لإضافة نص اتصال إلى الإعدادات بطريقة مرئية اتبع الخطوات التالية:

- افتح متصفح المشاريع Solution Explorer، وانقر مرتين بالفأرة فوق العنصر My Project.. سيؤدي هذا إلى فتح نافذة خصائص المشروع.
- اضغط العنصر Settings من الهامش الأيسر لفتح صفحة مصمم الإعدادات كما تعلمنا من قبل.
- في العمود Name اكتب اسم خاصية الإعداد، ولتكن BooksConStr.
- في العمود Type اضغط زر الإسدال، ومن القائمة المنسدلة اختر العنصر الخاص (Connection String).. هذا سيغير نطاق خاصية الإعداد Scope ليصير على مستوى التطبيق Application.
- اضغط الزر الموجود في خانة القيمة Value.. سيعرض لك هذا مربع حوار خصائص الاتصال الذي استخدمناه من قبل لإنشاء اتصال من متصفح

الخوادم Server Explorer.. هذا يتيح لك تكوين نص الاتصال بطريقة مرئية سهلة.. حدد مزود البيانات وقاعدة البيانات واختيارات الحماية.. ولو أردت استخدام المزيد من مفاتيح نص الاتصال، فاضغط الزر Advanced.. سيعرض لك هذا النافذة التالية:




هذه النافذة تعرض خصائص نص الاتصال، وهي نفس الخصائص التي شرحناها في الفئة SqlConnectionStringBuilder.. ويمكنك تغيير القيم الافتراضية لهذه الخصائص، وكل خاصية ستغيرها ستظهر في نص الاتصال الذي سيتم تكوينه.

- اضغط OK لإغلاق النافذتين.. سيظهر نص الاتصال الذي تم تكوينه في الخانة Value.

- وإذا كنت تحتاج إلى هذا، يمكنك إضافة نصوص اتصال أخرى إلى الإعدادات، بالكتابة في الصفوف التالية، وبهذا تجمع في مكان واحد، كل نصوص الاتصال اللازمة للتعامل مع كل قواعد البيانات والخوادم التي تحتاجها في برنامجك، مما يسهل عليك تعديلها في أي لحظة.
- اضغط زر الحفظ من شريط الأدوات لحفظ هذه التغييرات في ملف إعدادات المشروع.

الآن، تم توليد خاصية اسمها BooksConStr في فئة الإعدادات Settings في النطاق My، وتستطيع قراءة قيمتها في أي وقت بمنتهى البساطة.. جرب مثلا:

**MsgBox(My.Settings.BooksConStr)**

لاحظ أنك لا تستطيع تغيير قيمة الخاصية BooksConStr لأنها معرفة للقراءة فقط، لكن ما زال بوسعك فتح مصمم الإعدادات في أي لحظة لتغيير قيمة نص الاتصال، أو فتح الملف app.config من متصفح المشاريع، وتحرير قيمة الخاصية BooksConStr التي ستجدها تحت المقطع <connectionStrings>، دون الحاجة إلى تغيير أي كود في برنامجك.

## فئة مقطع نصوص الاتصال
## ConnectionStringsSection Class

هذه الفئة موجودة في النطاق System.Configuration، وهي ترث الفئة ConfigurationSection التي تعرفنا عليها في كتاب برمجة الويندوز.
ولا جديد في هذه الفئة، سوى امتلاكها للخاصية التالية:

**نصوص الاتصال ConnectionStrings:**
تعيد مجموعة من النوع ConnectionStringSettingsCollection، التي ترث الفئة ConfigurationElementCollection، وكل عنصر من عناصر هذه المجموعة هو من نوع فئة إعدادات نص الاتصال ConnectionStringSettings Class، التي ستتعرف عليها بعد قليل.
وتتيح لك هذه المجموعة قراءة كل نصوص الاتصال الموجودة في ملف الإعدادات.

ويمكنك استخدام إحدى وسائل فتح التهيئة OpenxxConfiguration الخاصة بمدير التهيئة ConfigurationManager للحصول على كائن تهيئة Configuration Object يتعامل مع النوع المراد من الإعدادات، ثم استخدام الخاصية ConnectionStrings لكائن التهيئة للحصول على نسخة من الفئة ConnectionStringsSection كالتالي:

```
Dim Cnfg = ConfigurationManager.OpenMachineConfiguration
Dim CnStrSett = Cnfg.ConnectionStrings
```

## فئة إعدادات نص الاتصال
## ConnectionStringSettings Class

هذه الفئة ترث فئة عنصر التهيئة ConfigurationElement Class، التي تعرفنا عليها في كتاب برمجة نماذج الويندوز.
ولحدث إنشاء هذه الفئة ثلاث صيغ:
١- الأولى بدون معاملات.
٢- والثانية تستقبل معاملين: اسم خاصية الإعداد التي ستحفظ نص الاتصال، ونص الاتصال نفسه.
٣- والثالثة تزيد على الصيغة السابقة بمعامل ثالث، يستقبل اسم مزود البيانات Provider الذي سيستخدم نص الاتصال.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

### الاسم Name:
تقرأ أو تغير اسم خاصية الإعداد التي ستحفظ نص الاتصال.

### نص الاتصال ConnectionString:
تقرأ أو تغير نص الاتصال المحفوظ في خاصية الإعداد.

### اسم المزود ProviderName:
تقرأ أو تغير اسم مزود البيانات الذي سيستخدم نص الاتصال.

ويمكنك الحصول على مجموعة إعدادات نصوص الاتصال الخاصة بالتطبيق، باستخدام الخاصية المشتركة Shared Property التالية:

```
Dim CnStrSett = ConfigurationManager.ConnectionStrings
```

والمثال التالي يعرض لك كل نصوص الاتصال الموجودة في ملف إعداد التطبيق:

```
Dim CnStrSett = ConfigurationManager.ConnectionStrings
For Each CnStr As ConnectionStringSettings In CnStrSett
    MsgBox(CnStr.Name)
    MsgBox(CnStr.ProviderName)
    MsgBox(CnStr.ConnectionString)
Next
```

| ملحوظة: |
| --- |
| يجب عليك حماية نص الاتصال بتشفيره، وذلك لأن ملف الإعدادات يتم توزيعه مع البرنامج، مما يجعل المستخدمين قادرين على قراءته وأخذ كلمات المرور منه.<br>ويمكنك تشفير مقطع نصوص الاتصال <ConnectionStrings> في ملف الإعدادات، بنفس الطريقة التي شرحناها في كتاب برمجة الويندوز، واستخدمناها في البرنامج AddAppSettings المرفق بذلك الكتاب. |

## واجهة الاتصال بقواعد البيانات

## IDbConnection Interface

هذه الواجهة تمثل الواجهة IDisposable، وهي تتيح لك إنشاء كائن اتصال خاص بك، وذلك بكتابة فئة تمثلها Implements the interface.. هذا يسهل عليك كتابة مزود جديد للتعامل مع نوع معين من قواعد البيانات غير متاح في إطار العمل.

وتمتلك الواجهة IDbConnection الخصائص التالية:

### نص الاتصال ConnectionString:

تقرأ أو تغير نص الاتصال الذي يحتوي على المعلومات اللازمة للاتصال بقاعدة البيانات.. ولا يمكنك تغيير قيمة هذه الخاصية إلا عندما يكون الاتصال مع قاعدة البيانات مغلقا.

### وقت الانتظار ConnectionTimeout:

تقبل عددا صحيحا، يمثل الوقت بالثانية، الذي سيتم انتظاره أثناء محاولة الاتصال بقاعدة البيانات، فإذا مر هذا الوقت دون أن يستجيب الخادم، يتم إلغاء العملية وينطلق خطأ في برنامجك.. والقيمة الافتراضية لهذه الخاصية هي ١٥ ثانية، لكن إذا أردت أن تظل منتظرا إتمام الاتصال إلى ما لانهاية، فضع صفرا في هذه الخاصية!.. لكن هذا قد يؤدي إلى توقف برنامجك عن العمل إذا فشلت عملية الاتصال بالخادم، لهذا لو استخدمت هذه القيمة فيجب أن تعطي للمستخدم طريقة لإلغاء محاولة الاتصال بنفسه، كأن تضع على النموذج زر إلغاء، مع جعل عملية الاتصال في عملية فرعية مستقلة Thread لكي لا يتوقف البرنامج عن الاستجابة.

لاحظ أن وضع قيمة كبيرة في هذه الخاصية سيؤدي إلى تعطيل البرنامج لفترة أطول، ووضع قيمة صغيرة فيها سيؤدي إلى فشل محاولات الاتصال بسرعة.. وأفضل قيمة لهذه الخاصية هي ما تراه مناسبا لظروف برنامجك.. فلو كنت تتوقع ضغطا كبيرا على الخادم يجعل استجابته لمحاولات الاتصال بطيئة أو متأخرة، فضع قيمة أكبر في هذه الخاصية (مثل ٦٠ أو ٩٠ مثلا).

### قاعدة البيانات Database:

تعيد اسم قاعدة البيانات التي يتم الاتصال بها.

## فئة اتصال سيكيول SqlConnection Class

هذه الفئة ترث الفئة DbConnection، مما يعني أنها ضمنيا ترث الفئة Component وتمثل الواجهة IDbConnection.
وبالإضافة إلى الخصائص التي ترثها من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

### إطلاق حدث الخطأ FireInfoMessageEventOnUserErrors:

إذا جعلت قيمة هذه الخاصية True، فسيتم إطلاق الحدث InfoMessage فور حدوث خطأ في الاتصال، ودون انتظار انتهاء تنفيذ الإجراء الذي أنشأ الاتصال.. أما إذا تركت قيمتها الافتراضية False، فسينطلق استثناء Exception في البرنامج عند حدوث خطأ في الاتصال، ولم ينطلق الحدث InfoMessage إلا بعد انتهاء تنفيذ الإجراء الذي أنشأ الاتصال.

### حجم حزم البيانات PacketSize:

تعيد حجم حِزم البيانات (بالوحدة الثنائية Byte) المستخدمة في نقل البيانات.

### تفعيل الإحصائيات StatisticsEnabled:

إذا جعلت قيمة هذه الخاصية True، فسيتم جمع إحصائيات عن عملية الاتصال.. لاحظ أن هذا مفيد في بعض الحالات، لكنه قد يؤدي إلى إبطاء الاتصال، لهذا لا تستخدمه إلا للضرورة.. والقيمة الافتراضية لهذه الخاصية هي False.

### معرف الجهاز WorkstationId:

تعيد اسم جهاز العميل المتصل بالخادم.

وبالإضافة إلى ما تمثله من وسائل الواجهة IDbConnection، تمتلك هذه الفئة الوسائل التالية:

### تغيير كلمة السر ChangePassword:

أرسل إلى هذه الوسيلة نص الاتصال، وكلمة السر الجديدة التي تريد استخدامها مع المستخدم المحدد في نص الاتصال بدلا من كلمة السر القديمة.. هذا معناه أن نص الاتصال يجب أن يحتوي على اسم المستخدم UserID وكلمة السر القديمة Password، لهذا لو أرسلت نص اتصال فيه خيار الحماية المتكاملة IntegratedSecurity فسيحدث خطأ.

وتقوم هذه الوسيلة بفتح اتصال خاص بها لتغيير كلمة السر، وإغلاقه فور الانتهاء من هذا، دون التعامل مع رصيد الاتصالات المساهمة Connection Pool.

وتفيدك هذه الوسيلة إذا كانت كلمة السر الخاصة بالمستخدم قد انتهت صلاحيتها Expired ويجب تغييرها.. ويمكنك معرفة هذا عند استخدام الوسيلة Open لفتح الاتصال، حيث سيحدث خطأ في البرنامج من النوع SqlException، وعليك أن تفحص قيمة الخاصية Number الخاصة بهذا الاستثناء، فإن وجدت قيمتها ١٨٤٨٧ أو ١٨٤٨٨ فهذا معناه انتهاء صلاحية كلمة السر ووجوب تغييرها.

والمثال التالي يحاول الاتصال بالخادم، فإن فشل الاتصال بسبب انتهاء صلاحية كلمة السر، فإنه يغير كلمة السر القديمة:

```
Dim Csb As New SqlConnectionStringBuilder
Csb.DataSource = ".\SQLEXPRESS"
Csb.InitialCatalog = "Books"
Csb.UserID = "User1"
Csb.Password = "2009"
Dim Cn As New SqlConnection(Csb.ToString)
Try
   Cn.Open()
Catch ex As SqlException
   If ex.Number = 18487 OrElse ex.Number = 18488 Then
      SqlConnection.ChangePassword(Csb.ToString, "2010")
   End If
End Try
```

**إلغاء المساهمة ClearPool:**

أرسل إلى هذه الوسيلة كائن الاتصال SqlConnection Object لإلغاء الاتصال الذي يمثله من رصيد الاتصالات المساهمة Connection Pool.. لاحظ أن هذا الاتصال قد يكون مستخدما في تلك اللحظة لأداء بعض الاستعلامات، لهذا يتم إنهاؤه في الحال، وسيظل مستخدما إلى حين إغلاقه باستخدام الوسيلة Close، وعندها لن يعود إلى رصيد الاتصالات المساهمة، بل سيغلق في الحال.. مثال:

**SqlConnection.ClearPool(Cn)**

**إلغاء كل أرصدة المساهمة ClearAllPools:**

تغلق جميع الاتصالات الموجودة في رصيد الاتصالات المساهمة Connection Pool، وإذا كان بعضها مستخدما، لا يتم إنهاؤه إلى أن يتم استدعاء الوسيلة Close الخاصة به.. مثال:

**SqlConnection.ClearAllPools( )**

**إضافة إلى قائمة التعاملات المنتشرة EnlistDistributedTransaction:**

مماثلة للوسيلة EnlistTransaction.

**الحصول على الإحصائيات RetrieveStatistics:**

تعيد مجموعة تمثل واجهة القاموس IDictionary، تحتوي على أزواج من المفاتيح Keys والقيم Values، تمثل إحصائيات الاتصال حتى هذه اللحظة.. ويمكنك استدعاء هذه الوسيلة أكثر من مرة على فترات، للحصول على أحدث قيم للإحصائيات.. لاحظ أنك لن تحصل على أي إحصائيات إلا إذا جعلت للخاصية StatisticsEnabled القيمة True أولا.

**تصفير الإحصائيات ResetStatistics:**

تعيد جميع قيم الإحصائيات إلى الصفر.

كما تمتلك هذه الفئة الحدث التالي:

## رسالة المعلومات InfoMessage:

ينطلق عندما يرسل الخادم رسالة تحذير أو خطأ.. والمعامل الثاني e لهذا الحدث من النوع SqlInfoMessageEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | Errors | تعيد مجموعة من النوع SqlErrorCollection، التي تمثل الواجهة ICollection، وكل عنصر من عناصرها من نوع الفئة SqlError، التي تحتوي على أحد الأخطاء أو التحذيرات التي أرسلها خادم سيكويل.. وسنتعرف على الفئة SqlError بعد قليل. |
| | Message | تعيد نصا يشرح أول خطأ موجود في مجموعة الأخطاء Errors.. هذا مفيد إن كان هناك خطأ واحد فقط. |
| | Source | تعيد نصا يحدد اسم الكائن الذي تسبب في أول خطأ موجود في مجموعة الأخطاء Errors.. هذا مفيد إن كان هناك خطأ واحد فقط. |

وقد استخدمنا فئة اتصال سيكويل في التطبيق AuthorBooks_Reader للاتصال بقاعدة بيانات الكتب على الخادم المحلي.. لاحظ أننا فتحنا الاتصال في حدث تحميل النموذج Load ولم نغلقه إلا في حدث إغلاق النموذج FormClosing، مما أتاح لنا استخدام نفس الاتصال لتنفيذ جميع الاستعلامات التي يقوم بها المستخدم.. ورغم أن كائن الاتصال معرف كمتغير موضعي Local Variable في حدث تحميل النموذج، إلا أننا وضعنا مرجعا له في الخاصية Connection الخاصة بكائن الأمر Command Object المعرف على مستوى النموذج، مما جعل كائن الاتصال حيا طالما كان كائن الأمر حيا.. هذا هو السبب في أننا استخدمنا الخاصية Connection الخاصة بكائن الأمر لإغلاق الاتصال في حدث إغلاق النموذج كالتالي:

**Cmd.Connection.Close( )**

لاحظ أن ترك كائن الاتصال مفتوحا طوال الوقت عمليّ فقط في حالتنا هذه، لأننا نتعامل هنا مع خادم محلي، وهناك نسخة واحدة فقط من البرنامج تتعامل معه.. لكن في البرامج العملية التي تتعامل مع خادم حقيقي، قد يؤدي ترك الاتصال مفتوحا إلى تقليل كفاءة البرنامج، خاصة إذا كان هناك عدد كبير من المستخدمين يتعاملون مع برنامجك في نفس اللحظة.

افترض مثلا أنك تصمم برنامجا لشركة يعمل بها ٢٥٠ موظفا، وأن خادم سيكويل مجهز لاستقبال ١٠٠ اتصال فقط في نفس اللحظة.. في هذه الحالة لو جعلت برنامجك يفتح نفس الاتصال بصورة دائمة طوال تشغيل المستخدم له، فإن أول ١٠٠ موظف يفتحون البرنامج على أجهزتهم سيمنعون خادم سيكويل من الاستجابة لمئة وخمسين موظفا آخرين إلى أن يغلق بعض المستخدمين البرنامج!
لقد أحببت أن أريك كيف يمكن أن يؤدي التصميم الخاطئ لبرنامجك إلى نتائج كارثية، ويعطل العمل ولا تجني من ورائه سوى السخط ☺ !
ورغم أن هذا مثال افتراضي لتقريب الفكرة، حيث إن سيكيول سيرفر يستطيع فعليا خدمة بضع مئات من العملاء وربما أكثر في نفس الوقت، إلا أن هذا العدد مهما بدا كبيرا لك فهو محدود، ويمكن تجاوزه عمليا في المؤسسات الضخمة كالحكومة الالكترونية مثلا (هل تتخيل كم عدد الموظفين في الوزارات المختلفة الذين يتعاملون مع قاعدة البيانات في نفس اللحظة؟)، أو في مواقع الإنترنت التي تحفظ قواعد بياناتها على سيكويل سيرفر، وأنت تعرف أن بعض هذه المواقع يصل زوارها إلى عدة ملايين يوميا، مما يؤدي إلى بطء استجابة الخادم، واعتذاره للكثير من العملاء عن وجود ضغط كبير بجبرهم على الانتظار (لا ريب أنك واجهت هذه المشكلة مع خادم بريد هوتميل في بعض الأوقات)!
ولحل هذه المشكلة في برنامجنا، عليك أن تنقل الكود من حدث تحميل النموذج وحدث إغلاقه إلى حدث ضغط الزر، ليتم فتح الاتصال وإغلاقه فقط عند الحاجة.. ولا تقلق من كثرة فتح برنامجك للاتصال وإغلاقه، فخاصية مساهمة الاتصالات Connection Pooling التي يدعمها خادم سيكويل تغنيك عن أي عناء لحل هذه المشكلة، حيث يتم ترك بعض الاتصالات مفتوحة لضمان سرعة الاستجابة للاستعلامات المتكررة، دون إعاقة بعض المستخدمين عن الاتصال بالخادم.. ولحسن الحظ فإن تقنية المساهمة Pooling تكون فعالة في الوضع الافتراضي، ما لم تطلب أنت إيقافها صراحة عبر نص الاتصال كما عرفنا سابقا.
وستجد كود الاتصال المحسّن في المشروع المسمى AuthorBooks_Reader2.

## فئة خطأ سيكويل SqlError Class

تحتوي هذه الفئة على معلومات على رسالة الخطأ أو التحذير التي أرسلها خادم سيكويل.. وتمتلك هذه الفئة الخصائص التالية:

**الرتبة Class:**
تعيد رقما من ٠ إلى ٢٥٥ يمثل درجة خطورة الخطأ.. وقيمتها الافتراضية ٠.

**الخادم Server:**
تعيد نصا يمثل اسم نسخة سيكويل سيرفر التي أرسلت الخطأ.

**المصدر Source:**
تعيد اسم المزود Provider الذي تسبب في الخطأ.

**الإجراء Procedure:**
تعيد اسم الإجراء المخزن الذي تسبب في الخطأ.

**رقم السطر LineNumber:**
تعيد رقم السطر الذي تسبب في الخطأ في الإجراء المخزن.

**الرسالة Message:**
تعيد نصا يصف رسالة الخطأ.. ويمكنك أيضا استخدام الوسيلة ToString الخاصة بهذا الكائن لعرض نص هذه الرسالة.

**الرقم Number:**
تعيد رقم الخطأ.

**الحالة State:**
تعيد رقما من ٠ إلى ٢٥٥ يمثل الكود الرقمي للخطأ.

| **ملحوظة:** |
| --- |
| بالإضافة إلى ما ترثه فئة استثناء سيكويل SqlException من خصائص ووسائل من الفئة الأم SystemException والفئة الأم Exception، فإنها تمتلك نفس خصائص الفئة SqlError، مما يمنحك القدرة على الحصول على نفس المعلومات، سواء استخدمت الحدث InfoMessage أم استخدمت الطريقة التقليدية لمعالجة الأخطاء Exception Handling. |

# كـائـــن الأمـــــر

# Command Object

يتعامل كائن الأمر مع استعلام SQL أو إجراء مخزن Stored Procedure، مع امتلاكه الوسائل اللازمة لتنفيذهما عبر اتصال مفتوح، واستلام النتيجة من الخادم. وسنتعرف في هذا الفصل على كيفية التعامل مع كائن الأمر.

## واجهة أمر قاعدة البيانات IDbCommand Interface

هذه الواجهة تمثل الواجهة IDisposable، وهي توجد في النطاق System.Data.. وتمتلك هذه الفئة الخصائص التالية:

**الاتصال Connection:**

تستقبل هذه الخاصية أي كائن اتصال يمثل الواجهة IDbConnection، ليتم استخدامه في تنفيذ الأمر.. ويشترط فتح الاتصال أولا قبل محاولة تنفيذ الأمر ، وإلا حدث خطأ.

**نوع الأمر CommandType:**

تحدد نوع الأمر المراد تنفيذه، وهي تأخذ إحدى قيم المرقم CommandType التالية:

| | |
|---|---|
| Text | الأمر يتكون من جملة SQL.. وهي القيمة الافتراضية. |
| StoredProcedure | الأمر يتكون من اسم إجراء مخزن يراد تنفيذه. |
| TableDirect | الأمر يتكون من اسم جدول يراد تحميل كل بياناته كاملة من قاعدة البيانات مباشرة.. هذه القيمة غير مقبولة مع قواعد بيانات سيكويل سيرفر، لكن يمكن استخدامها مع قواعد بيانات آكسيس. |

### نص الأمر CommandText:

جملة SQL التي تريد تنفيذها، أو اسم الإجراء المخزن الذي تريد تنفيذه، أو اسم الجدول المراد تحميله، وذلك تبعا لقيمة الخاصية CommandType.
لاحظ أنك تستطيع كتابة أكثر من جملة SQL في هذه الخاصية مع الفصل بينها بالفاصلة المنقوطة ; ، وفي هذه الحالة سيتم تنفيذها جميعا، والحصول على أكثر من مجموعة من النتائج Resultsets، وهو نفس ما يمكن أن يحدث عند استدعاء إجراء مخزن يقوم بتنفيذ أكثر من جملة SELECT.. وسنعرف كيف يمكن التعامل مع هذه النتائج لاحقا.

### وقت انتظار الأمر CommandTimeout:

تحدد وقت الانتظار بالثانية، الذي سيتم بعده اعتبار محاولة تنفيذ الأمر فاشلة.. والقيمة الافتراضية لهذه الخاصية هي ٣٠ ثانية، وعليك أن تزيدها إذا كانت الاستعلام معقدا يتم على جداول كثيرة ويجري عليها الكثير من العمليات، أو كان هناك ضغط كبير على الخادم يجعل استجابته بطيئة.

### المعاملات Parameters:

تعيد أي كائن يمثل واجهة "مجموعة معاملات البيانات" IDataParameterCollection، التي ترث واجهة القائمة IList.. وتتيح لك مجموعة المعاملات إضافة المعاملات المطلوب التعويض بها في جملة الاستعلام أو الإجراء المخزن.. وسنتعرف على المعاملات بالتفصيل لاحقا.

### التعامل Transaction:

تستقبل أي كائن من نوع واجهة تعاملات قاعدة البيانات IDbTransaction، ليتم تنفيذ الأمر الحالي في نطاقه.

### مصدر الصفوف المحدّثة UpdatedRowSource:

تحدد كيف سيتم تحديث صفف البيانات DataRow الموجود في أحد جداول مجموعة البيانات DataSet.. لاحظ أن هذه الخاصية مفيدة فقط عندما يكون كائن الأمر الحالي هو أمر التحديث Update Command الخاص بموصل البيانات DbDataAdapter الذي يملأ مجموعة البيانات ويحدثها.. ويتم تنفيذ أمر التحديث على الصفوف التي تغيرت في مجموعة البيانات واحدا تلو الآخر، وقد يحتوي هذا الأمر على معاملات إخراج Output Parameters، أو ترافقه جملة SELECT تعيد السجل بعد تحديثه في قاعدة البيانات.. الحكمة في هذا أن هناك بعض القيم التي تولدها قاعدة

البيانات بنفسها (مثل عمود الترقيم التلقائي) ولا يمكنك معرفة قيمة هذه الأعمدة إلا بالحصول على السجل مرة أخرى بعد تحديثه (أو إضافته).. وتتحكم هذه الخاصية في كيفية الاستفادة من القيم العائدة من أمر التحديث، وهي تأخذ إحدى قيم المرقم UpdateRowSource التالية:

| | |
|---|---|
| None | يتم تجاهل أي معاملات أو صفوف عائدة من أمر التحديث. |
| OutputParameters | توضع قيم معاملات الإخراج Output Parameters في خانات سجل مجموعة البيانات DataSet. |
| FirstReturnedRecord | توضع قيم أول سجل عائد من أمر التحديث، في سجل مجموعة البيانات. |
| Both | توضع قيم معاملات الإخراج وأول سجل عائد من أمر التحديث، في سجل مجموعة البيانات. |

كما تمتلك هذه الواجهة الوسائل التالية:

- **تجهيز Prepare:**

تنشئ نسخة محسنة مجهزة من الأمر وتحفظها على الخادم، ليكون تنفيذها أسرع.. ولا تستقبل هذه الوسيلة معاملات، وليس لها قيمة عائدة، ولا يكون لها أي تأثير إن كانت للخاصية CommandType القيمة TableDirect.
لاحظ أن استدعاء هذه الوسيلة صار عديم القيمة تقريبا، لأن إصدارات سيكويل سيرفر ٢٠٠٠ و ٢٠٠٥ و ٢٠٠٨ تقوم بتجهيز البيانات تلقائيا عند اللزوم، لتحسين الأداء.

- **إنشاء معامل CreateParameter:**

تعيد كائنا من النوع IDbDataParameter، لتخصصه للتعامل مع أحد المعاملات الموجودة في الاستعلام.

- **تنفيذ بدون استعلام ExecuteNonQuery:**

تنفذ الأمر دون أن تعيد أية سجلات، ولكن تعيد عددا صحيحا Integer يمثل عدد السجلات التي تأثرت بتنفيذ الأمر.. ويمكنك استخدام هذه الوسيلة لتنفيذ أوامر التحديث Update والحذف Delete والإدراج Insert.
وستجد مثالا على هذه الوسيلة في الزر CREATE PROC في المشروع AccessStoredProcedure الذي عرفنا من قبل أنه ينشئ إجراء مخزنا في قاعدة بيانات الكتب المنشأة بتطبيق Access.. في هذا الزر نستخدم

كائن أمر من النوع OleDbCommand للتعامل مع قاعدة بيانات Access، ونستخدم الوسيلة ExecuteNonQuery لتنفيذ استعلام SQL الذي ينشئ الإجراء المخزن في قاعدة البيانات، لأنه لا يعيد إلينا أي ناتج.
لاحظ أن كود الاتصال بقاعدة البيانات وتنفيذ الأوامر يتكرر كثيرا، وهو يبدو طويلا مع وجود تعديلات طفيفة.. لهذا سيكون من الأذكى لو جمعنا الكود المتشابه في إجراء وأرسلنا إليه المعاملات التي تناسب الاستعلام الذي ننفذه.. ولقد فعلنا هذا في المشروع DbTasks، حيث عرفنا فيه فئة اسمها MyDbConnector، وأضفنا إليها خاصية اسمها ConnectionStr تستقبل نص الاتصال، وحدث إنشاء Constuctor يستقبل نص الاتصال أيضا ويضعه في هذه الخاصية على سبيل الاختصار.. وقد عرفنا في هذه الفئة عددا من الوسائل التي تتيح لنا التعامل مع قاعدة البيانات، ومن بينها دالة اسمها ExcuteCommand.. هذه الدالة تستقبل معاملين:

- نص الأمر CommandText الذي تريد تنفيذه.. ويمكنك أن ترسل إلى هذا المعامل استعلام SQL أو اسم إجراء مخزن، حيث سنقوم باستنتاج نوع الأمر بحيلة صغيرة، فلو كان النص يبدأ بأي من أوامر SQL مثل " insert" أو " update"... إلخ، فمعنى هذا أنه نوع الأمر CommandType.Text.. لاحظ أننا وضعنا مسافة بعد اسم الكلمة، تلافيا لاحتمال أن تكون هذه الكلمة جزءا من اسم إجراء مخزن (مثل UpdateAuthors)، فنحن واثقون أن اسم الإجراء المخزن لا يحتوي على مسافات، بينما الاستعلامات تحتوي على مسافات.. غير هذا يكون نوع الأمر CommandType.StroresProcedure.
- مصفوفة نصية ثنائية البعد String(,)، يتكون كل عنصر فيها من اسم المعامل وقيمته.. وبهذا يمكنك تمرير المعاملات مباشرة إلى الدالة، حيث سنضيف هذه المعاملات إلى مجموعة معاملات الأمر Command.Parameters.. وإذا لم يكن للأمر معاملات، فأرسل إلى المعامل الثاني للدالة Nothing.

وتستخدم هذه الدالة الوسيلة Command.ExecuteNonQuery لتنفيذ الأمر، وتعيد True إذا لم يحدث خطأ، وتعيد False إذا حدث خطأ.
وستجد مثالا على استخدام هذه الدالة في نفس المشروع، حيث وضعنا مربعي نص على النموذج لاستقبال اسم المؤلف ونبذة عنه، وعند ضغط الزر يتم تنفيذ استعلام لإضافته إلى جدول الكتب.. انظر كيف سيكون الكود في منتهى البساطة والاختصار باستخدام الفئة MyDbConnector:

```
Dim DbBooks As New MyDbConnector(
                    My.Settings.BooksConStr)
Dim SQL = "INSERT INTO Authors " &
          " (Author, CountryID, About)" &
          " VALUES (@Author, 21, @About)"
Dim Params = {{"@Author", TxtAuthor.Text.Trim},
              {"@About", TxtAbout.Text.Trim}}
If DbBooks.ExcuteCommand(SQL, Params) Then
   TxtAuthor.Clear()
   TxtAbout.Clear()
End If
```

ويمكنك استخدام الفئة MyDbConnector للتعامل مع أي قاعدة بيانات، وتنفيذ أي أمر عليها باستخدام الإجراء ExcuteCommand.. أليس هذا شيئا مريحا؟

### تنفيذ قارئ ExecuteReader:

تنفذ الأمر، وتعيد كائنا من النوع IDataReader، حيث يمكنك استخدامه لقراءة النتيجة سجلا تلو سجلّ.. وسنتعرف على قارئ البيانات لاحقا.

وستجد مثالا على هذه الوسيلة في المشروع AuthorBooks_Reader.

ولهذه الوسيلة صيغة ثانية، تستقبل معاملا من نوع المرقم CommandBehavior، الذي يحدد سلوك قارئ البيانات، كالتالي:

| | |
|---|---|
| Default | السلوك العادي، حيث يمكن أن يؤدي تنفيذ الأمر إلى الحصول على أكثر من مجموعة من مجموعات النتائج Result Sets (كما يحدث في حالة تنفيذ أكثر من جملة SQL من داخل إجراء مخزن).. هذا مكافئ لاستدعاء الوسيلة ExecuteReader بدون معاملات. |
| SingleResult | يؤدي تنفيذ الاستعلام إلى الحصول على مجموعة نتائج واحدة فقط. |
| SchemaOnly | يؤدي تنفيذ الاستعلام إلى الحصول على معلومات الأعمدة فقط بدون أي سجلات.. هذا يعني الحصول على جدول فارغ به أسماء الأعمدة فقط.. هذا مماثل لاستخدام مزود سيكويل للخيار:<br>SET FMTONLY ON |

| | |
|---|---|
| KeyInfo | يؤدي تنفيذ الاستعلام إلى الحصول على معلومات الأعمدة والمفتاح الأساسي Primary Key. |
| SingleRow | يؤدي تنفيذ الاستعلام إلى الحصول على سجل واحد فقط، ولو كان الاستعلام يحصل على أكثر من مجموعة من النتائج، فسيكون بكل مجموعة منها سجلا واحدا فقط.. استخدم هذا الاختيار عندما تحتاج إلى أو سجل فقط، فهذا يؤدي إلى تحسين أداء وسرعة البرنامج. |
| Sequential Access | قراءة تتابعية.. هذا مفيد للقراءة من الأعمدة التي تحوي قدرا ضخما من البيانات، فبدلا من طلبها كلها من الخادم، يتم طلب أجزاء من البيانات فقط تبعا لاحتياجك.. لاحظ الآتي:<br>– يجب عليك قراءة قيم الحقول بنفس ترتيبها في الاستعلام، لأنك لو قرأت أي حقل، فلن تستطيع قراءة الحقل السابق له مرة أخرى، فنحن هنا نقرأ البيانات تتابعيا، أي بالترتيب.<br>– استخدم الوسيلة GetValue لقراءة القيمة الموجودة في أي حقل كاملة.<br>– استخدام الوسيلة GetBytes الخاصة بقارئ البيانات لقراءة أجزاء من الحقل الذي يحتوي على بيانات ثنائية ضخمة، مثل image و varbinary(MAX).<br>– استخدام الوسيلة GetChars الخاصة بقارئ البيانات لقراءة أجزاء من الحقل الذي يحتوي على نصوص ضخمة، مثل text و ntext و varchar(MAX) و nvarchar(MAX).<br>والمشروع ReadLargeData يقرأ الصور الخاصة بشعار كل ناشر من الجدول Publishers، ويحفظها في ملف على الجهاز.. ونظرا لأن الصورة قد تكون ضخمة، فقد استخدمنا هذا الخيار لنقرأ البيانات تتابعيا.. ويريك هذا المشروع أن هذه الطريقة تصلح للقراءة من الحقل Logo الذي نوعه image، وتصلح أيضا للقراءة من الحقل Logo2 الذي نوعه varbinary(MAX). |
| Close Connection | يتم إغلاق الاتصال Connection مع قاعدة البيانات آليا، بمجرد إغلاق قارئ البيانات. |

ويمكنك استخدام أكثر من قيمة من هذه القيم، بربطها معا باستخدام المعامل Or.

وقد أضفنا وسيلة اسمها GetReader إلى الفئة MyDbConnector في المشروع DbTasks، مهمتها تنفيذ استعلام باستخدام الوسيلة ExecuteReader وإعادة قارئ البيانات.. لاحظ أنك لو أغلقت الاتصال في نهاية هذه الوسيلة فسيحدث خطأ عند محاولتك استخدام قارئ البيانات الذي أعادته إليك، لأن الاتصال الذي يستخدمه قد تم إغلاقه.. لهذا عليك عدم إغلاق الاتصال، وإرسال القيمة CommandBehavior.CloseConnection إلى معامل الوسيلة ExecuteReader لجعل قارئ البيانات يغلق الاتصال بنفسه عندما يتم إغلاقه.. وقد جعلنا للوسيلة GetReader معاملا اختياريا اسمه Sequential إذا جعلته True فستحصل على قارئ بيانات تتابعي لاستخدامه في قراءة البيانات الضخمة على أجزاء، والقيمة الافتراضية لهذا المعامل هي False لتحصل على قارئ بيانات عادي.
وستجد مثالا لاستخدام هذه الوسيلة في نفس المشروع في زر "الكتب".. هذا الزر يعرض كتب المؤلف الذي كتبت اسمه في مربع النص العلوي.

### تنفيذ قيمة ExecuteScalar:

تنفذ الأمر، وتعيد كائنا Object يحتوي على قيمة الخانة الموجودة في الصف الأول من العمود الأول في الجدول الناتج، وتتجاهل باقي الخانات.
ويمكنك استخدام هذه الوسيلة لتنفيذ دوال التجميع Aggregate Functions.
والمشروع AvgPrice يريك مثالا على استخدام هذه الوسيلة لمعرفة متوسط أسعار الكتب.
وقد أضفنا وسيلة اسمها GetValue إلى الفئة MyDbConnector في المشروع DbTasks، مهمتها تنفيذ استعلام باستخدام الوسيلة ExecuteScalar وإعادة القيمة الناتجة، وستجد مثالا لاستخدامها في نفس المشروع في الزر "معرفة عدد المؤلفين".

### إلغاء Cancel:

تحاول إلغاء تنفيذ الأمر.. ولا يحدث خطأ إن لم يكن الأمر قيد التنفيذ حاليا، أو إن فشلت في إيقاف تنفيذه.

## فئة أمر قاعدة البيانات DbCommand Class

هذه الفئة أساسية مجردة Abstract Base Class، تجب وراثتها MustInherit، وهي ترث فئة المكون Component Class، كما أنها تمثل الواجهة IDbCommand وبالتالي تمتلك جميع وسائلها وخصائصها.
وبالإضافة إلى ما تمثله من خصائص الواجهة IDbCommand، تمتلك هذه الفئة الخاصية التالية:

### مرئية في وقت التصميم DesignTimeVisible:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيظهر كائن الأمر في وقت التصميم في واجهة الأدوات التي تستخدمه.

ولا تمتلك هذه الفئة أي وسائل جديدة غير ما تمثله من وسائل الواجهة IDbCommand.
لاحظ أن الفئات التالية ترث الفئة DbCommand:

١. OdbcCommand Class.
٢. OleDbCommand Class.
٣. SqlCommand Class.
٤. OracleCommand Class.

وسنكتفي هنا بالتعرف على الفئة SqlCommand.

## فئة أمر سيكويل SqlCommand Class

هذه الفئة ترث الفئة DbCommand، وهي مخصصة للتعامل مع الأوامر التي يتم تنفيذها على خادم سيكويل سيرفر.
ولحدث إنشاء هذه الفئة أربع صيغ مختلفة:

١. الصيغة الأولى بدون معاملات.
٢. والصيغة الثانية لها معامل واحد، يستقبل نص الاستعلام الذي سيوضع في الخاصية CommandText.
٣. والصيغة الثالثة تزيد على الصيغة السابقة بمعامل ثان من النوع SqlConnection، يستقبل كائن الاتصال الذي سيتم تنفيذ الأمر من خلاله.
٤. والصيغة الأخيرة تزيد على الصيغة السابقة بمعامل ثالث من النوع SqlTransaction، يستقبل كائن التعامل الذي سيتم تنفيذ الأمر في نطاقه.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخاصيتين التاليتين:

**التنبيه Notification:**
تحدد كائن طلب التنبيه SqlNotificationRequest الذي سيستخدمه كائن الأمر في تلقي التنبيهات من الخادم عند تنفيذ الاستعلام.. وسنتعرف على الفئة SqlNotificationRequest لاحقا.

**ضم تلقائي إلى قائمة التنبيهات NotificationAutoEnlist:**
إذا جعلت قيمة هذه الخاصية True، فسيستقبل كائن الأمر تنبيهات تلقائية من كائن معلومات التبعية SqlDependency.. وتستخدم هذه الخاصية مع صفحات المواقع في ASP.NET لتتيح عرض الصفحة المجهزة Cashed Page إلى أن يأتي تنبيه بحدوث تغير في بعض بياناتها فيتم إنعاشها.. وسنتعرف على الفئة SqlDependency لاحقا.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الوسائل التالية:

**نسخ Clone:**
تعيد كائن SqlCommand جديدا مماثلا في كل شيء للكائن الحالي.

**تصفير زمن انتظار الأمر ResetCommandTimeout:**
تعيد قيمة الخاصية CommandTimeout إلى قيمتها الافتراضية ٣٠ ثانية.

**"تنفيذ قارئ بيانات XML" ExecuteXmlReader:**
تنفذ الأمر، وتعيد كائنا من النوع XMLReader، الذي يحتوي على البيانات بتنسيق XML.. لاحظ أن عليك استخدام هذه الوسيلة في الحالات التالية:

- عند استخدام الفقرة FOR XML في جملة الاستعلام.. هذه الفقرة تعيد ناتج الاستعلام في صورة وثيقة XML.
- عند قراءة عمود نوع بياناته XML.
- عند قراءة عمود نوع بياناته ntext أو nvarchar لكنه يحتوي على بيانات بتنسيق XML.

ولن نتطرق إلى فئات XML في هذا الكتاب، وسنفرد لها كتابا مستقلا إن قدر الله.

وتدعم الفئة SqlCommand استخدام العمليات غير المتزامنة Asynchronous Operations لتنفيذ الأمر، وذلك من خلال أزواج الوسائل التالية:

| | |
|---|---|
| **BeginExecuteXmlReader** | **EndExecuteXmlReader** |
| **BeginExecuteNonQuery** | **EndExecuteNonQuery** |
| **BeginExecuteReader** | **EndExecuteReader** |

حيث تقوم الوسائل التي تبدأ بالكلمة Begin بتنفيذ الأمر في عملية غير متزامنة، ويمكنك أن ترسل إليها مندوبا عن إجراء يتم استدعاؤه بعد انتهاء العملية لتقرأ فيه البيانات الناتجة.. وتعيد هذه الوسائل كائنا يمثل الواجهة IAsyncResult ليمكنك استخدامه في متابعة العملية، كما يمكن إرساله إلى الوسائل التي تبدأ بالكلمة End لإنهاء العملية غير المتزامنة والحصول على نتائجها.
وتمتاز العمليات غير المتزامنة بأنها لا توقف تنفيذ البرنامج إلى حين انتهاء إتمام العملية، بل ينتقل التنفيذ إلى السطر التالي مباشرة، بينما ترسل النتائج فور توفرها إلى الدالة الخاصة بالحصول على النتائج Callback Function.. وتقع العمليات غير المتزامنة خارج نطاق هذا الكتاب، وسنتعرف عليها بالتفصيل بإذن الله في كتاب المواضيع المتقدمة في برمجة إطار العمل.

كما تمتلك الفئة SqlCommand الحدث التالي:

### اكتملت الجملة StatementCompleted:

ينطلق عند اكتمال تنفيذ جملة الاستعلام الخاصة بكائن الأمر.. والمعامل الثاني e لهذا الحدث من النوع StatementCompletedEventArgs، وهو يمتلك خاصية واحدة هي "عدد السجلات" RecordCount، التي تعيد عدد السجلات التي تأثرت بتنفيذ جملة الاستعلام.

## تمرير القيم إلى جمل الاستعلام:

افترض أنك تريد الحصول على كتب "توفيق الحكيم" من قاعدة البيانات.. في هذه الحالة يمكنك وضع جملة SQL التالية في الخاصية CommandText لكائن الأمر (وليكن اسمه Cmd):

```
Cmd.CommandText = "SELECT Books.Book " +
    "FROM Authors, Books " +
    "WHERE Authors.ID = AuthorID " +
    "AND Authors.Author = 'توفيق الحكيم'"
```

ولكن، هل تظن أنك ستكتب الجملة السابقة في أي تطبيق عملي فعلاً؟.. هل ستقتصر وظيفة برنامجك على عرض كتب مؤلف واحد فقط، أم أنك ستسمح للمستخدم باختيار المؤلف الذي يريده ليعرض له البرنامج كتب هذا المؤلف؟
المنطقي والعملي، هو أن تضع على النموذج مربع نص (وليكن اسمه TxtAuthor) ليكتب فيه المستخدم اسم المؤلف، ومن ثم تعرض له كتبه.. في مثل هذه الحالة، عليك تعديل نص الاستعلام السابق ليصير كالتالي:

```
Cmd.CommandText = "SELECT Books.Book " +
    "FROM Authors, Books " +
    "WHERE Authors.ID = AuthorID " +
    "AND Authors.Author = '" + TxtAuthor.Text +"'"
```

حيث استخدمنا طريقة تشبيك النصوص في الكود، لإضافة النص الموجود في مربع النص إلى جملة الاستعلام، وبهذا حصلنا على جملة استعلام مرنة، تستطيع البحث عن اسم أي مؤلف يريده المستخدم.
لكن هذه الطريقة تحتوي على ثغرة قاتلة، تسمح للأشقياء بتدمير قاعدة بياناتك وربما نظام التشغيل الذي يوجد عليه خادم سيكويل لو أرادوا!
كيف؟.. هذا هو موضوع الفقرة التالية.

## دس الاستعلامات SQL Injection:

في المثال السابق، سمحنا للمستخدم بكتابة اسم المؤلف في مربع نص، ثم أدرجنا محتوى النص داخل جملة الاستعلام كجزء من شرط الفقـرة WHERE.. وقـد تفتق ذهن بعض العباقرة عن فكرة شريرة، وهي كتابة بعض جمل الاستعلام في مربع النص بدلا من اسم المؤلف، وبهذا يستطيعون حقن اسـتعلامات مدسوسـة خاصة بهم داخل جملة الاستعلام الخاصة بك، فيقوم برنامجك بتنفيذ ما يريـدون على قاعدة البيانات، وهو أمر يشبه سلوك الفيروسات التي تحقن مادتها الوراثيـة في نواة الخلية وتتركها تنفذها لإنتاج فيروسات جديدة!

وخطورة هذه الطريقة، هي أنها تتيح للمخترقين الاستعلام عن بعـض حسابات المديرين والمستخدمين، ومعرفة تركيب الجداول، بل وتنفيذ بعض أوامر غـلاف الويندوز Shell من خلال خادم سكيويل، مما قد يضر بالجهاز الذي يعمل عليـه الخادم!

ولكن كيف تتم عملية الحقن Injection؟

١- أول شيء، يتوقع القرصان Hacker ضرورة وجود النص بين علامتي تنصيص، ولا بد أنك وضعت علامة تنصيص بادئة قبل الـنص الـذي سيأتي من مربع النص، لهذا يجب على المخترق أن يكتب أي كلمة، ثـم يتبعها بالعلامة ' لإغلاق علامتي التنصيص، وبهذا يضمن عدم حـدوث خطأ في صيغة جملة SQL.

٢- بعد هذا يضع القرصان فاصلة منقوطة ; ليستطيع كتابة أمر SQL جديد خاص به، وهنا تكون لديه الحرية في كتابة الأمر الذي يريده!

٣- نظرا لأن القرصان يتوقع منك إضافة تكملة لجملة SQL بعد النص الذي كتبه في مربع النص، فإنه يضع في نهاية الكود المدسـوس الرمـز -- ليجعل أي نص تال له مجرد تعليق، وبهذا يلغي أي تكملة خاصـة بـك لجملة الاستعلام، ويضمن سلامة صيغة جملة الاستعلام!

والآن، دعنا نرى ماذا سيحدث لو كتب القرصان في مربع النص الجملة التالية:

```
Ahamd'; drop table Books--
```

في هذه الحالة ستصبح جملة الاستعلام بعد إضافة هذه الجملة كالتالي:

```
SELECT Books.Book
FROM Authors, Books
WHERE Authors.ID = AuthorID
AND Authors.Author = 'Ahamd'; drop table Books--'
```

كما ترى: صار لدينا استعلامان صحيحان وتعليق:

- الاستعلام الأول لا قيمة له، وهو يبحث عن كتب مؤلف اسمه Ahmad.

- والاستعلام الثاني أمر حذف يطلب حذف جدول الكتب كاملا من قاعدة البيانات.. لاحظ أن القرصان لا يعرف أسماء الجداول، ولكنّ توقع اسم جدول الكتب لن يكون عسيرا، ولن ييأس القرصان من تجربة عشرات الأسماء المحتملة، ما دام عزمه قد قرّ على تدمير برنامجك!
- وفي النهاية يوجد تعليق صغير، هو العلامة ' الخاصة بك، والتي استطاع القرصان تهميشها بحيلة صغيرة بارعة!

يبدو الأمر مفزعا، أليس كذلك؟

إن هذه الثغرة تتيح للقراصنة تدمير قاعدة البيانات، ودخول حسابات المستخدمين، دون الحاجة إلى كتابة اسم المستخدم أو كلمة المرور، والكثير من الكوارث التي تكفي لتطير النوم من عيون المبرمجين ☺.

فكيف إذن يمكن إغلاق هذه الثغرة القاتلة؟

في الحقيقة هناك عدة نصائح هامة في هذا الصدد:

١- التقليل من استخدام مربعات النص، والاستعاضة عنها بأدوات تتيح اختيار القيم، مثل القوائم Lists إن كان هذا ممكنا.

٢- استخدام الخاصية MaxLength الخاصة بمربع النص لتحديد طول النص المسموح بكتابته في مربع النص.. هذا سيحد من قدرة القرصان على كتابة أوامر مدسوسة.

٣- إجراء بعض الفحوصات الصغيرة على قيمة مربع النص، للتأكد من خلو النص الذي كتبه المستخدم من العلامات المريبة مثل ; ' -- */ /* .. هذا سيشل حركة القرصان تماما.. والأفضل أن تمنع كتابة هذه الحروف في مربع النص من المنبع باستخدام الحدث KeyPress.

٤- عليك أيضا أن تمنع الكلمات الدالة على أوامر SQL في مربع النص، خاصة DROP و DELETE و UPDATE و INSERT.

٥- لا تقبل أيا من الكلمات التالية في مربع نص يدخل فيه المستخدم اسم ملف:

AUX, CLOCK$, CON, CONFIG$, NUL, PRN
COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8
LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8

٦- من المهم أيضا أن تحدد صلاحيات مستخدمي قاعدة البيانات، وألا تعطي الصلاحيات الخطيرة (كحذف الجداول أو إنشائها) إلا للمديرين، وعليك أن تصنع نسخة خاصة من البرنامج لهؤلاء المديرين بحيث لا يتم تداولها إلا بينهم.. أما المستخدمون العاديون، فعليك أن تصنع لهم نسخة أخرى من البرنامج، وأن تتصل هذه النسخة بالخادم من خلال حساب مستخدم محدود الصلاحيات، وبهذا لو نجح أي قرصان في تجاوز خطوط دفاعك عبر هذه النسخة، لا يجد الكثير مما يستطيع فعله!

٧- كن حذرا من الكلمات التي تبدأ بـ xp_، لأنها البادئة التي يتم بها تسمية الإجراءات المخزنة الإضافية لمخطط قاعدة البيانات Catalog-extended stored procedures، مثل الإجراء xp_cmdshell.

٨- استخدام الإجراءات المخزنة Stored Procedures في تنفيذ الاستعلامات، لأنها تكون محفوظة في قاعدة البيانات على الخادم، ومن ثم يقوم سيكويل سيرفر بفحص قيم المعاملات المرسلة إلى الإجراء المخزن، والتأكد من أنها من النوع الصحيح وبالطول المحدد.

٩- ارفض كتابة الكلمات التالية في مربع النص، لأنها تسمح بحقن الأكواد المدسوسة في الإجراءات المخزنة:

EXECUTE, EXEC, sp_executesql

١٠- استخدام المعاملات Parameters لتمرير القيم إلى جمل الاستعلام، بدلا من استخدام طريقة تشبيك النصوص Concatination، لأن المعاملات تضمن التحقق من نوع المعامل والطول المسموح به لقيمته.. لكن نصيحة: لا تتخل عن فحص النصوص التي يكتبها المستخدم في مربعات النصوص، حتى لو استخدمت المعاملات، فقد تسمح بعض المعاملات النصية الطويلة بعبور بعض الكود المدسوس.

وفي المشروع AuthorBooks_Reader استخدمنا الإجراء SqlInjection للتأكد من أن النص الذي كتبه المستخدم في مربع النص لا يحتوي على أية رموز أو كلمات مريبة، كما استخدمنا معاملا لتمرير النص إلى الاستعلام لمزيد من الحماية.

كما استخدمنا الدالة SqlInjection في المشروع DbTasks للتأكد من أن قيم المعاملات لا تحتوي على استعلامات مدسوسة.

وسنتعرف فيما يلي على المعاملات وكيفية استخدامها.

## المعاملات Parameters:

المعامل هو علامة موضعية Placeholder توضع في جملة SQL لتشير إلى أن هناك قيمة سيتم التعويض بها بدلا منها.. ويمكنك تعريف أي عدد تريده من المعاملات في جملة الاستعلام.

وتختلف صيغة هذه العلامة تبعا لنوع مزود البيانات، فمزود سيكويل يستخدم الرمز @ لتمييز المعامل، يتبعه اسم متغير بدون أن يفصل بينهما مسافات (مثل @UserName).. لهذا نستطيع كتابة الاستعلام عن كتب أحد المؤلفين كالتالي:

```
Cmd.CommandText = "SELECT Books.Book " +
        "FROM Authors, Books " +
        "WHERE Authors.ID = AuthorID " +
        "AND Authors.Author = @Author"
```

هذا يبدو كأننا عرفنا متغيرا اسمه Author@ واستخدمناه في جملة الاستعلام، ليتم التعويض عنه عند تنفيذها.

أما في مزود OLEDB و ODBC فيتم استخدام علامة الاستفهام الإنجليزية ? للإشارة إلى وجود معامل، دون منح هذا المعامل أي اسم:

```
Cmd.CommandText = "SELECT Books.Book " +
        "FROM Authors, Books " +
        "WHERE Authors.ID = AuthorID " +
        "AND Authors.Author = ?"
```

لاحظ أن قواعد بيانات آكسيس صارت تقبل تسمية المعاملات (وما زالت تقبل العلامة ? أيضا)، لكنها لا تميز المعاملات المسماة باستخدام أي علامة خاصة.. يمكنك مثلا أن تكتب الاستعلام السابق كما يلي:

```
Cmd.CommandText = "SELECT Books.Book " +
        "FROM Authors, Books " +
        "WHERE Authors.ID = AuthorID " +
        "AND Authors.Author = AuthorValue"
```

في هذه الحالة ستعتبر آكسيس أن AuthorValue هو اسم معامل.. بل يمكنك أيضا أن تستخدم الاسم Author@ في الاستعلام، وستقبله آكسيس كاسم معامل، وهذا يساعدك على استخدام نفس استعلامات سيكويل سيرفر مع آكسيس!

لكن هذا المرونة من آكسيس تسبب مشكلة غريبة في بعض الأحيان، فلو أخطأت مثلا في كتابة اسم الحقل AuthorID في الاستعلام السابق، وكتبته مثلا AutherID، فستعتبره آكسيس اسم معامل، وبدلا من أن تحصل في برنامجك على رسالة تخبرك أن هذا العمود ليس موجودا في الجدول، ستحصل على رسالة تخبرك بأن قيم بعض المعاملات مفقودة.. هذا هو السبب الذي سيجعلك ترى آلاف الأسئلة من المبرمجين عن سبب ظهور هذه الرسالة الغريبة في برنامجهم:

No value given for one or more required parameters"

رغم أنهم لا يستخدمون استعلامات فيها معاملات، أو أنهم مرروا قيم المعاملات الصحيحة فعلا!.. فكل ما هناك، أنهم أخطأوا في كتابة اسم أحد الحقول، فتم اعتباره معاملا!
ولكن، كيف يمكن التعويض عن قيم المعاملات؟
لفعل هذا، عليك استخدام مجموعة المعاملات الخاصة بكائن الأمر DbCommand.Parameters لتعريف كائنات المعاملات ووضع القيم فيها، حيث سيقوم كائن الأمر بتمريرها إلى جملة الاستعلام عند تنفيذ الأمر.. وعليك أن تنتبه جيدا إلى أن وضع رمز المعامل في جملة الاستعلام لا ينشئ معاملات في مجموعة المعاملات تلقائيا، فهذا الرمز يحدد فقط موضع التعويض عن المعامل، بينما تظل أنت مسئولا عن تعريف معامل في مجموعة المعاملات لتحديد نوع القيمة التي يقبلها المعامل، وتمرير القيمة من خلاله إلى كائن الأمر.
ويشترط في حالة سيكويل سيرفر أن يكون لكل من المعامل الموجود في نص الاستعلام والمعامل الموجود في مجموعة المعاملات نفس الاسم.. فإذا عرفت في نص الاستعلام معاملا اسمه Author@، فيجب أن يكون اسمه في مجموعة المعاملات أيضا Author@.
أما معاملات OLEDB و ODBC فكلها ممثلة بالرمز ؟ مما يجعلك مضطرا إلى تعريفها في مجموعة المعاملات DbCommand.Parameters بنفس ترتيب ظهورها في نص الاستعلام.. وحتى لو كنت تستخدم معاملات مسماة في آكسيس، فما زلت مضطرا إلى المحافظة على الترتيب الصحيح لهذه المعاملات، كما أن اسم كل معامل ما زال غير مهم، لهذا تستطيع تعريف معامل في مجموعة المعاملات اسمه X ليعوض عن معامل في نص الاستعلام اسمه Author@.. فكل ما يهم حقا هو الترتيب وليس الاسم.. يمكنك اعتبار أن آكسيس يمحو اسم المعامل ويضع بدلا منه علامة استفهام ؟ على سبيل التسهيل عليك.
هذا هو ما جعل من الممكن أن نستخدم نفس جملة الاستعلام، ونفس كود تعريف المعاملات في حدث ضغط زر "الكتب" في المشروع Factories للحصول على كتب أحد المؤلفين من قاعدة الكتب في آكسيس أو قاعدة الكتب في سيكويل سيرفر.
والآن، دعنا نتعرف على مجموعة المعاملات والفئات المستخدمة معها.

## فئة مجموعة معاملات قاعدة البيانات
## DbParameterCollection Class

هذه المجموعة تمثل الواجهة IDataParameterCollection، التي ترث واجهة القائمة IList.. وكل عنصر يضاف إلى المجموعة DbParameterCollection هو من نوع الفئة DbParameter التي سنتعرف عليها لاحقا.

ولا تحتوي هذه الفئة على أية خصائص أو وسائل جديدة غير ما تمثله من عناصر الواجهة.

لاحظ أن الفئة DbParameterCollection أساسية مجردة تجب وراثتها، لهذا ترثها كل من الفئات التالية:

١. OdbcParameterCollection Class
٢. OleDbParameterCollection Class
٣. SqlParameterCollection Class
٤. OracleParameterCollection Class

ويعنينا هنا أن نتعرف على فئة مجموعة معاملات سيكويل SqlParameterCollection Class.

## فئة مجموعة معاملات سيكويل
## SqlParameterCollection Class

هذه الفئة ترث الفئة DbParameterCollection، وهي لا تختلف عنها كثيرا إلا في أن عناصرها من نوع الفئة SqlParameter ، التي سنتعرف عليها بعد قليل. كما أن للوسيلة Add الخاصة بهذه الفئة العديد من الصيغ التي من المفيد أن نتعرف عليها:

### إضافة Add:

تضيف معاملا إلى مجموعة المعاملات، ولها الصيغ التالية:

١- الصيغة الأولى تستقبل معاملا واحدا من النوع SqlParameter.

٢- الصيغة الثانية تستقبل نصا يمثل اسم المعامل وإحدى قيم المرقم SqlDbType التي توضح نوع المعامل.

٣- الصيغة الثالثة تزيد على الصيغة السابقة بمعامل ثالث من النوع Integer، يستقبل حجم البيانات التي ستوضع في المعامل.
٤- الصيغة الرابعة تزيد على الصيغة السابقة بمعامل، يستقبل اسم عمود في مجموعة البيانات DataSet ليربط المعامل به.
٥- وهناك صيغة خامسة لكنها لم يعد من المنصوح استخدامها، تستقبل اسم المعامل، وكائنا Object يحمل قيمته، وذلك بسبب التعارض بينها وبين الصيغة الثانية، لهذا تم إضافة وسيلة جديدة اسمها AddWithValue كبديل لهذه الصيغة.

### إضافة بالقيمة AddWithValue:

تضيف معاملا إلى المجموعة، وهي تستقبل اسم المعامل، وكائنا Object يحمل قيمته.. وتعيد هذه الوسيلة مرجعا إلى المعامل الذي تم إنشاؤه.
لاحظ أنك تستطيع أن ترسل إلى المعامل الثاني لهذه الوسيلة قارئ بيانات DataReader أو جدول بيانات DataTable، لتتم قراءة كل الصفوف الموجودة فيهما ووضعها في المعامل.. هذا مفيد إذا كنت تتعامل مع إجراء مخزن يستقبل معاملا جدولا Table-Valued Parameter، وتريد أن ترسل إليه جدولا كاملا.. وستجد مثالا على هذا في المشروع TableValuedParameters.. في هذا المشروع نقرأ جدول المؤلفين في قارئ بيانات من النوع OledbDataReader، ثم نرسله كمعامل ثان إلى الوسيلة AddWithValue لاستخدامه كمعامل للإجراء المخزن InsertAuthors، وبهذا نستطيع إضافة المؤلفين من قاعدة بيانات آكسيس إلى قاعدة بيانات سيكويل سيرفر.
لاحظ أن قارئ البيانات يجب أن يظل مفتوحا هو والاتصال الذي يستخدمه، لأن الوسيلة AddWithValue لا تنسخ السجلات من قارئ البيانات فعليا، ولا يتم نسخ هذه السجلات إلا عند استدعاء الوسيلة ExecuteNonQuery الخاصة بكائن الأمر الذي ينفذ الإجراء المخزن ويرسل إله المعامل الجدول.

## واجهة معامل البيانات IDataParameter Interface

هذه الواجهة معرفة في النطاق System.Data، وهي تمتلك الخصائص التالية:

### اسم المعامل ParameterName:

تحدد اسم المعامل، مع ملاحظة أنه يبدأ بالرمز @، مثل UserName@.

### هل هو منعدم IsNullable:

إذا جعلت قيمة هذه الخاصية True، فسيقبل المعامل القيمة DBNull.. والقيمة الافتراضية هي False.

### الاتجاه Direction:

تحدد اتجاه المعامل، وهي تأخذ إحدى قيم المرقم ParameterDirection التالية:

| | |
|---|---|
| Input | معامل إدخال، لإرسال قيمة إلى الاستعلام أو الإجراء المخزن. |
| Output | معامل إخراج، لقراءة قيمة متغير إخراج من الإجراء المخزن.. هذا النوع غر متاح مع قواعد بيانات آكسيس! |
| InputOutput | معامل إدخال وإخراج معا. |
| ReturnValue | قيمة عائدة من إجراء مخزن. |

### القيمة Value:

تقرأ أو تغير قيمة المعامل، وهي من النوع Object لتقبل أي نوع من القيم، لكن هذا لا يعني أن كل القيم مسموح بها، لأن الخاصية DbType تحدد نوع القيم المسموح بها.

ويجب وضع القيمة في هذه الخاصية قبل تنفيذ الاستعلام من خلال كائن الأمر، وإذا كان المعامل للإخراج، فإن القيمة العائدة من الخادم توضع في هذه الخاصية بعد تنفيذ الأمر أو بعد إغلاق قارئ البيانات DataReader إن كنت تستخدمه.

ولوضع القيمة DBNull في هذه الخاصية، استخدم الفئة DBNull كالتالي:

**P.Value = DBNull.Value**

حيث إن Value هي خاصية مشتركة للقراءة فقط Shared ReadOnly Property معرفة في الفئة DBNull لتعيد نسخة جديدة من هذه الفئة تمثل القيمة DBNull.

## النوع DbType:

تحدد نوع المعامل، وهي تأخذ إحدى قيم المرقم DbType التالية:

| الثابت | معناه | مجال القيم أو طول المتغير |
|---|---|---|
| SByte | وحدة ثنائية بإشارة. | من -١٢٨ إلى ١٢٧ |
| Byte | وحدة ثنائية موجبة. | من ٠<br>إلى ٢٥٥ |
| Binary | بيانات ثنائية | من ١ بايت<br>إلى ٨٠٠٠ بايت |
| Boolean | قيمة منطقية | True أو False |
| Int16 | عدد قصير | من -٣٢٧٦٨<br>إلى ٣٢٧٦٧ |
| UInt16 | عدد قصير موجب | من ٠ إلى ٦٥٥٣٥ |
| Int32 | عدد صحيح | من -٢١٤٧٤٨٣٦٤٨<br>إلى ٢١٤٧٤٨٣٦٤٧ |
| UInt32 | عدد صحيح موجب | من ٠<br>إلى ٤٢٩٤٩٦٧٢٩٥ |
| Int64 | عدد طويل | من -<br>٩٢٢٣٣٧٢٠٣٦٨٥٤٧٧٥٨٠٨<br>إلى ٩٢٢٣٣٧٢٠٣٦٨٥٤٧٧٥٨٠٧ |
| UInt64 | عدد طويل موجب | من ٠ إلى<br>١٨٤٤٦٧٤٤٠٧٣٧٠٩٥٥١٦١٥ |
| Single | عدد مفرد | من ١,٥ × (١٠ ^ -٤٥)<br>إلى ٣,٤ × (١٠^٣٨)<br>بدقة ٧ خانات عشرية. |
| Double | عدد مزدوج | من ٥,٠ × (١٠^-٣٢٤)<br>إلى ١,٧ × (١٠^٣٠٨)<br>بدقة ١٥ خانة عشرية. |
| Currency | عملة | من -٢ ^ ٦٣ إلى (٢ ^ ٦٣) -١<br>بدقة ٤ خانات عشرية. |

| الثابت | معناه | مجال القيم أو طول المتغير |
|---|---|---|
| Decimal | عدد عشري | من ١,٠ × (١٠^-٢٨)<br>إلى ٧,٩ × (١٠^٢٨)<br>بدقة تصل إلى ٢٨ خانة عشرية. |
| VarNumeric | رقم متغير | قيمة رقمية متغيرة الطول، تقبل أيا من الأنواع السابقة. |
| Guid | معـــرف عـــام متفـــرد | |
| Time | وقت | وقت بدون تاريخ |
| Date | تاريخ | تاريخ بدون وقت |
| DateTime | تاريخ ووقت | تقبل تواريخ بين ١٧٥٣/١/١ و ٩٩٩٩/١٢/٣١ |
| DateTime2 | تاريخ ووقت | تقبل تواريخ بين ١/١/١ و ٩٩٩٩/١٢/٣١ |
| DateTime Offset | إزاحة الوقت والتاريخ | |
| String | نص | نص متغير الطول، مكون من حروف موسعة Unicode. |
| StringFixed Length | نص ثابت الطول | نص ثابت الطول، مكون من حروف موسعة Unicode. |
| AnsiString | نص قياسي | نص متغير الطول، مكون من حروف قياسية بترميز ASCII. |
| AnsiString Fixed Length | نص قياسي ثابت الطول | نص ثابت الطول، مكون من حروف قياسية بترميز ASCII. |
| Xml | XML | صفحة XML أو جزء منها. |
| Object | كائن | أي نوع غير موجود في الأنواع السابقة. |

### عمود المصدر SourceColumn:

تحدد اسم العمود في مجموعة البيانات DataSet الذي سيتم ربطه بالمعامل.. هذا مفيد عند حفظ التغيرات من مجموعة البيانات إلى قاعدة البيانات باستخدام أمر التحديث Update Command.. هذه هي الخطوات التي تحدث:

- يعرّف أمر التحديث Update Comman معاملا لكل عمود في مجموعة البيانات، ويضع اسم العمود في الخاصية SourceColumn لكل معامل.
- عند تنفيذ عملية التحديث، يقوم موصل البيانات DataAdapter بالمرور على صفوف مجموعة البيانات واحدا تلو الآخر وتنفيذ أمر التحديث على كل منها على حدة.
- لتحديث أي صف، يضع موصل البيانات في كل معامل من معاملات أمر التحديث، قيمة الخانة الموجودة في العمود المحدد في الخاصية DbParameter.SourceColumn، وبهذا يمكن تنفيذ استعلام التحديث لكل صف بصورة صحيحة.

وستفهم هذه الأمور بصورة أوضح عندما نتعرف على موصل البيانات DataAdapter ومجموعة البيانات DataSet.

### إصدار المصدر SourceVersion:

تحدد نوع القيمة التي ستوضع في المعامل.. هذا مفيد عند استخدام المعامل لتحديث مجموعة البيانات، لأن كل خانة في مجموعة البيانات تحتفظ بعدة أنواع من القيم، مثل القيمة الأصلية (القادمة من قاعدة البيانات)، والقيمة الحالية (القيمة الجديدة التي أدخلها المستخدم).. وتأخذ هذه الخاصية إحدى قيم المرقم DataRowVersion.. وسنتعرف على هذا الأمر بتفصيل أكبر لاحقا عند التعرف على مجموعة البيانات.

## ⊶ واجهة معامل بيانات قاعدة البيانات
## IDbDataParameter Interface

هذه الواجهة ترث الواجهة IDataParameter، وهي تمتلك بعض الخصائص الإضافية التي تضيف مزيدا من التحكم في القيم التي يقبلها المعامل.. وهذه الخصائص هي:

### الحجم Size:

تحدد أقصى حجم مسموح به للمعامل بالوحدة الثنائية Byte.. والقيمة الافتراضية لهذه الخاصية تستنتج من نوع المعامل، فإن كان عددا صحيحا على سبيل المثال، تكون قيمتها ٤، وإن كان المعامل يحتوي على مصفوفة فإن هذه الخاصية تأخذ طول المصفوفة.. لاحظ أنك لو صغرت قيمة هذه الخاصية عن حجم البيانات، فسيتم أخذ جزء من هذه البيانات فقط وإسقاط الجزء الزائد.

### الدقة Precision:

تحدد أكبر عدد مسموح به من الخانات الرقمية في القيمة التي يقبلها المعامل.. والقيمة الافتراضية هي ٠، وهي تعني عدم فرض قيود على عدد الخانات، وترك ذلك لمزود قاعدة البيانات.

### المقياس Scale:

تحدد أكبر عدد مسموح به من الخانات العشرية في القيمة التي يقبلها المعامل.. ولو زاد عدد الخانات عن هذا الرقم يتم تقريبه.. والقيمة الافتراضية هي ٠ .

## فئة معامل قاعدة البيانات DbParameter Class

هذه الفئة معرفة في النطاق System.Data.Common، وهي فئة أساسية مجردة تجب وراثتها، تمثل الواجهة IDbDataParameter مما يعني أنها تمثل أيضا الواجهة IDataParameter، وبالتالي فهي تملك كل خصائصهما، ولا تزيد عليها إلا خاصية واحدة جديدة وهي:

### تمثيل قيمة منعدمة في عمود المصدر SourceColumnNullMapping:

إذا جعلت قيمة هذه الخاصية True، فسيعني هذا أن هذا المعامل يستخدم لإخبارك إن كان عمود المصدر SourceColumn في مجموعة البيانات فارغا أم لا، حيث تكون قيمة المعامل ٠ إن كان العمود فارغا، وتكون قيمته ١ إن كان العمود يحتوي على أي قيمة.. هذا مفيد عند تعريف استعلامات التحديث، لأن مقارنة أي خانتين قيمتهما Null تكون نتيجته False رغم أن الخانتين متساويتين فعلا!.. لهذا يجب أن نتأكد قبل إجراء المقارنة إن كانت الخانتين فارغتين أم لا.. وسنرى كيف نستخدم هذه الطريقة عند التعرف على موصل البيانات في فصل لاحق.

كما تمتلك هذه الفئة وسيلة واحدة، وهي:

### تصفير النوع ResetDbType:

تعيد الخاصية DbType إلى قيمتها الافتراضية.

والفئات التالية ترث هذه الفئة:

١. OdbcParameter Class
٢. OleDbParameter Class
٣. SqlParameter Class
٤. OracleParameter Class

وسنقتصر هنا على التعرف على الفئة SqlParameter.

## فئة معامل سيكويل SqlParameter Class

هذه الفئة ترث الفئة DbParameter، وهي تمثل معامل استعلام موجه إلى سيكويل سيرفر.

ولحدث إنشاء هذه الفئة سبع صيغ، الصيغة الأولى بدون معاملات، والصيغ الأخرى تتيح لك إمداد المعامل ببعض قيم خصائصه، مثل اسم المعامل ونوعه واتجاهه.. إلخ.. على سبيل المثال، تستقبل الصيغة السابعة ١٣ معاملا هي بالترتيب:

- اسم المعامل.
- إحدى قيم المرقم SqlDbType توضح نوع المعامل.
- عدد صحيح يوضح طول المعامل إذا كان نصا أو وحدات ثنائية Bytes.. بالنسبة للمعاملات العددية استخدم القيمة صفر.. وإذا كان المعامل من الأنواع القصوى (MAX)، فاستخدم القيمة -١.
- إحدى قيم المرقم ParameterDirection التي توضح اتجاه المعامل.
- وحدة ثنائية Byte توضح عدد الخانات العشرية للمعامل الرقمي (الخاصية Precision).
- وحدة ثنائية Byte توضح عدد خانات التقريب العشري للمعامل الرقمي (الخاصية Scale).
- اسم عمود المصدر في مجموعة البيانات، الذي سيأخذ منه المعامل قيمته.
- إحدى قيم المرقم DataRowVersion، توضح إصدار العمود في مجموعة البيانات.. وسنتعرف على هذا المرقم في فصل لاحق.
- قيمة منطقية، إذا جعلتها True، فستكون وظيفة معامل البيانات أن يتأكد أن العمود ليس فارغا Null في مجموعة البيانات، حيث تكون قيمة المعامل ٠ إذا كان العمود فارغا، وتكون قيمته ١ إذا لم يكن فارغا.
- كائن Object يحمل القيمة التي تريد تمريرها إلى المعامل.. وعليك أن تستخدم القيمة Nothing، إذا كان المعامل سيقرأ القيمة من مجموعة البيانات.
- نص ترسل قيمته إلى الخاصية XmlSchemaCollectionDatabase التي سنتعرف عليها لاحقا.
- نص ترسل قيمته إلى الخاصية XmlSchemaCollectionOwningSchema التي سنتعرف عليها لاحقا.
- نص ترسل قيمته إلى الخاصية XmlSchemaCollectionName التي سنتعرف عليها لاحقا.

وهناك صيغة تستقبل فقط أول أربعة معاملات من الصيغة السابقة، وفي هذه الحالة يكون اتجاه المعامل للإدخال، ويقرأ القيمة الحالية للسجل Current Value.. وهناك صيغة أخرى تستقبل كل معاملات الصيغة السابقة ما عدا آخر ثلاثة معاملات.

والكود التالي يعرف معاملا يقرأ قيمته من الحقل Book في مجموعة البيانات:

```
Dim PrmBook As New SqlParameter("@Book",
    SqlDbType.NVarChar, 0, "Book")
```

والكود التالي يعرف معاملا يأخذ قيمته من النسخة الأصلية للحقل ID في مجموعة البيانات:

```
Dim PrmID As New SqlParameter("@Original_ID",
    SqlDbType.Int, 0, ParameterDirection.Input, False, 0,
    0, "ID", DataRowVersion.Original, Nothing)
```

ويمكنك استخدام هذين المعاملين لتعريف أمر التحديث Update كالتالي:

```
Dim UpdateCmd As New SqlCommand
UpdateCmd.CommandText = "UPDATE Books " & _
    "SET Book = @Book WHERE ID = @Original_ID"
UpdateCmd.Connection = Me.SqlConnection1
UpdateCmd.Parameters.AddRange(
    New SqlParameter( ) {PrmBook, PrmID})
```

ويمكنك جعل هذا الأمر أمر التحديث الخاص بموصل البيانات كالتالي:

```
DaAuthorBooks.UpdateCommand = UpdateCmd
```

وستجد هذا الكود في حدث تحميل النموذج Load في المشروع ViewAndEditBooks.. ويمكنك نقل التغييرات من مجموعة البيانات إلى قاعدة البيانات بضغط زر الحفظ، حيث يقوم موصل البيانات DaAuthorBooks بتنفيذ الوسيلة Update، التي تستخدم أمر التحديث والمعاملات التي عرفناها.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك الفئة SqlParameter الخصائص التالية:

### المعرف المحلي LocaleId:

تحدد معرف الثقافة المحلية للغة التي تريد استخدامها في التعامل مع قيمة المعامل.. على سبيل المثال: معرف اللغة العربية بالنظام السداسي عشري هو: 0x0001 . (راجع كتاب: "برمجة إطار العمل"، للمزيد من التفاصيل عن الثقافات العالمية ومعرفاتها).

### معلومات المقارنة CompareInfo:

تحدد كيف سيقوم هذا المعامل بمقارنة النصوص، وهي تأخذ إحدى قيم المرقم SqlCompareOptions التي تعرفنا عليها في الفصل السادس.

### الإزاحة Offset:

تحدد موضع بداية القراءة من الخاصية Value.. وتقاس الإزاحة بعدد الوحدات الثنائية Bytes عند التعامل مع بيانات ثنائية، وبعدد الحروف Characters عند التعامل مع نصوص.. والقيمة الافتراضية هي ٠، أي أن القراءة تتم من بداية البيانات.

وتحدد الخاصية Size عدد الحروف أو الوحدات الثنائية التي ستتم قراءتها بدءا من الموضع Offset.. هذا مفيد إذا وضعت في الخاصية Value كما كبيرا من البيانات، وأردت تجزئتها دون الحاجة إلى متغيرات وسيطة وحيل برمجية ملتوية.. في هذه الحالة ستعطي للخاصية Offset مبدئيا القيمة صفر وللخاصية Size الطول الذي تريده (وليكن ١٠٠).. بعد هذا تستخدم حلقة تكرار Loop تنفذ فيها الأمر على قاعدة البيانات (حيث سيقرأ الأمر الجزء المحدد فقط في المعامل بدءا من الموضع Offset وبالطول Size)، ومن ثم تزيد قيمة الخاصية Offset بمقدار ١٠٠ لقراءة جزء تال.. ويستمر الدوران وتنفيذ هذه العملية إلى أن تتجاوز طول البيانات الموجودة في الخاصية Value.. لاحظ أن هذه الطريقة مناسبة للاستخدام فقط مع الأمر Update.. وستجد مثالا على هذا في الزر Write .Parameter.Offset في المشروع Write Large Data، وهو يسمح لك بإضافة صورة شعار في العمود Logo2 للناشر الثاني.. لاحظ أننا حملنا كل محتويات ملف الصورة مرة واحدة ووضعناها في الخاصية Value للمعامل، ومن ثم أرسلناها إلى قاعدة البيانات على أجزاء صغيرة.. هذه الطريقة تحتوي على عيب كبير، وهو أن الصورة إذا كانت ضخمة جدا، فستستهلك مساحة كبيرة من الذاكرة وقد تسبب بطء البرنامج.. لهذا لا تستخدم هذه الطريقة إلا إذا كان حجم الصورة معقولا، أما إذا كان ضخما، فالأفضل أن تحمل أجزاء من الملف وترسلها إلى قاعدة البيانات بالطريقة التي استخدمناها في الزر Write .Update في نفس البرنامج.

## نوع بيانات سيكويل SqlDbType:

تحدد نوع المعامل من بين أنواع البيانات في سيكويل سيرفر، وهي تأخذ إحدى قيم المرقم SqlDbType، وهي تحمل نفس أسماء أنواع بيانات سيكويل سيرفر التي تعرفنا عليها بالتفصيل في الفصل الثالث.. والقيمة الافتراضية لهذه الخاصية هي NVarChar.

لاحظ أن هذه الخاصية مرتبطة بالخاصية DbType، لهذا لو غيرت قيمة إحداهما فستتغير الأخرى تلقائيا، بحيث تحتوي الخاصية SqlDbType دائما على أنسب نوع من أنواع سيكويل سيرفر يوافق نوع المتغير الموجود في الخاصية DbType.. جرب مثلا:

```
Dim P As New SqlParameter
P.SqlDbType = SqlDbType.Money
MsgBox(P.DbType.ToString("g")) ' Currency
P.DbType = DbType.Int64
MsgBox(P.SqlDbType.ToString("g")) ' BigInt
```

## اسم النوع TypeName:

إذا كنت تستخدم المعامل لإرسال قيمة إلى معامل جدول Table-Valued، فضع اسم هذا النوع في هذه الخاصية، متضمنا اسم المالك Owner.. فمثلا، للتعامل مع النوع AuthorType الذي يستخدمه الإجراء المخزن InsertAuthors، وضعنا في هذه الخاصية النص "dbo.AuthorType" أو باختصار "AuthorType" لأن المالك هنا افتراضي.. وفي هذه الحالة لا بد ان نضع في الخاصية SqlDbType القيمة SqlDbType.Structured.. وهذا هو ما فعلناه في المشروع TableValuedParameters.

## اسم المتغير الخاص بالمستخدم UdtTypeName:

إذا كنت تستخدم المعامل لإرسال قيمة إلى نوع من تعريف المستخدم User-Defined Type، فضع اسم هذا النوع في هذه الخاصية، ولا تنسَ أن تضع في الخاصية SqlDbType القيمة SqlDbType.Udt.. وسنتعرف على المتغيرات التي يعرفها المستخدم في سيكويل سيرفر لاحقا.

## قيمة سيكويل SqlValue:

مماثلة للخاصية Value، وكلتاهما تقرأ أو تغير قيمة المعامل.

### قاعدة بيانات المخطط XmlSchemaCollectionDatabase:

تعيد اسم قاعدة البيانات التي توجد بها مجموعة مخططات XML.. وإذا كانت قيمة هذه الخاصية نصا فارغا، فهذا معناه أن المخططات توجد في قاعدة البيانات الحالية، أو أنه لا توجد مخططات أصلا، وفي الحالة الأخيرة ستكون قيمة الخاصيتين XmlSchemaCollectionName و XmlSchemaCollectionOwningSchema نصا فارغا.

### اسم مجموعة المخططات XmlSchemaCollectionName:

تعيد اسم مجموعة مخططات XML.

### XmlSchemaCollectionOwningSchema:

تعيد مخطط العلاقات الرئيسي، الذي يحدد موضع مجموعة مخططات XML.

# قارئ البيانات DataReader

يتم إنشاء قارئ البيانات DataReader باستدعاء الوسيلة ExecuteReader الخاصة بكائن الأمر Command Object.. ويستقبل قارئ البيانات نتيجة الاستعلام الذي ينفذه كائن الأمر، ويقوم بتخزين ما يصل من البيانات من الخادم في المخزن الوسيط للشبكة Network Buffer الموجود على جهاز العميل، حيث يمكنك المرور عبر السجلات المستلمة واحدًا تلو الآخر على التوالي، مما يوفر ميزتين هامتين:

١- السرعة: حيث يمكنك قراءة السجلات المتوفرة فور وصولها، دون انتظار اكتمال وصول كل السجلات أولا.

٢- عدم استهلاك الذاكرة: لأن قارئ البيانات يحتفظ بسجل واحد فقط في الذاكرة في كل مرة.

لكنّ لهذه الطريقة عيبين أساسيين:

١- عدم القدرة على تحديث سجلات قاعدة البيانات.. بعبارة أخرى: قارئ البيانات للقراءة فقط كما يقول اسمه، وليس للكتابة!

٢- عدم القدرة على التراجع إلى الخلف، أو القفز مباشرة إلى سجل في موضع معين في النتيجة دون المرور على ما قبله من السجلات.

لهذا يوصف قارئ البيانات بأنه "مجرى بيانات للأمام فقط وللقراءة فقط":

Forward-only, Read-only Stream.

لكل هذا، يمكنك استخدام قارئ البيانات في الحالات التالية:

١- لو كنت ستتعامل مع سجلّ واحد فقط.

٢- لو كنت ستقرأ كل سجل مرة واحدة فقط، ولا يعنيك الرجوع إليه مرة أخرى.

٣- لو كانت قاعدة البيانات موجودة على نفس الجهاز، مما يعني سرعة الحصول على البيانات منها مباشرة، دون الحاجة إلى تحميلها في الذاكرة.

٤- عندما تريد قراءة النتائج دون الحاجة إلى تغيير أي جزء منها في قاعدة البيانات.

## واجهة سجل البيانات IDataRecord Interface

تقدم هذه الواجهة الخصائص والوسائل اللازمة لقراءة محتويات السجل الحالي في قارئ البيانات.
وتمتلك هذه الواجهة الخاصيتين التاليتين:

### عدد الحقول FieldCount:

تعيد عدد الأعمدة في السجل.. هذا يتيح لك كتابة حلقة تكرار Loop للمرور عبر كل الأعمدة بدءا من العمود رقم صفر إلى العمود رقم FieldCount - 1.. هذا مفيد لاختصار الكود عندما تستخدم استعلام يعيد عددا كبيرا من الحقول.

### العنصر Item:

هذه هي الخاصية الافتراضية Default Property، وهي تستقبل رقم العمود أو اسمه كمعامل، وتعيد كائنا Object يحتوي على القيمة الموجودة في السجل الحالي في هذا العمود.. والمثال التالي يعرض قيمة الخانة الاولى في الصف:

```
MsgBox(Reader.Item(0))
```

أو باختصار:

```
MsgBox(Reader(0))
```

كما تمتلك هذه الواجهة الوسائل التالية:

### معرفة الاسم GetName:

أرسل إلى هذه الوسيلة رقم العمود، لتعيد إليك اسمه.

### معرفة الترتيب GetOrdinal:

أرسل إلى هذه الوسيلة اسم العمود، لتعيد إليك رقمه.

### معرفة نوع الحقل GetFieldType:

أرسل إلى هذه الوسيلة رقم العمود، لتعيد إليك كائن النوع Type الذي يمثل نوع بياناته.

### معرفة اسم نوع البيانات GetDataTypeName:

أرسل إلى هذه الوسيلة رقم العمود، لتعيد إليك نصا يمثل اسم نوع بياناته.

### ❖ معرفة القيمة GetValue:

أرسل إلى هذه الوسيلة رقم العمود، لتعيد إليك كائنا Object يحتوي على القيمة الموجودة في السجل الحالي في هذا العمود.

### ❖ معرفة القيم GetValues:

أرسل إلى هذه الوسيلة مصفوفة كائنات Object Array، ليتم ملؤها بالبيانات الموجودة في خانات الصف الحالي في قارئ البيانات.. لاحظ أنك لو أرسلت مصفوفة أقصر من عدد خانات الصف الحالي فلن يحدث خطأ، بل سيتم نسخ جزء من الخانات فقط إلى المصفوفة وإهمال الباقي.. أما لو أرسلت مصفوفة أطول من عدد خانات الصف الحالي، فسيتم نسخ كل الخانات إلى جزء من المصفوفة وترك باقي المصفوفة فارغا.. وفي كل الأحوال، تعيد هذه الوسيلة عدد الخانات التي تم نسخها إلى المصفوفة.

### ❖ قراءة الوحدات الثنائية GetBytes:

تستخدم عندما يكون قارئ البيانات تتابعيا Sequential، وهي تعيد مصفوفة وحدات ثنائية Bytes، تحتوي على البيانات الموجودة في أحد الأعمدة.. ولهذه الوسيلة المعاملات التالية:

- رقم العمود.
- موضع بداية القراءة من محتويات العمود.
- مصفوفة وحدات ثنائية Bytes لاستقبال البيانات.. لاحظ أن خطأ سيحدث لو كانت المصفوفة أقصر من البيانات المطلوبة.
- موضع بداية الكتابة في المصفوفة.
- عدد الوحدات الثنائية Bytes المطلوب قراءتها من محتويات العمود، بدءا من الموضع المحدد في المعامل الثاني.. لاحظ أن خطأ سيحدث لو كان الطول المطلوب أكبر من البيانات المتبقية في العمود.

وتعيد هذه الوسيلة عدد الوحدات التي تم نسخها إلى المصفوفة، ولو أرسلت إلى هذه الوسيلة مصفوفة فارغة Nothing، فستعيد إليك العدد الإجمالي للوحدات الثنائية المتاحة في الخانة.

وقد استخدمنا هذه الوسيلة في المشروع ReadLargeData لقراءة بيانات الصورة، حيث نقرأ ١٠٠ وحدة ثنائية Byte من بداية الصورة ونحفظها في الملف، ثم نقرأ ١٠٠ وحدة تالية ونحفظها في الملف، ونستمر في فعل هذا إلى أن نكمل قراءة الصورة.. لاحظ أن شرط التوقف عن القراءة، هو أن تكون القيمة العائدة من الوسيلة GetBytes أصغر من عدد البيانات الذي طلبنا قراءته، مما يعني أن هذه هي آخر بيانات متاحة في الخانة.

### قراءة الحروف GetChars:

تستخدم عندما يكون قارئ البيانات تتابعيا Sequential، وهي تعيد مصفوفة حروف Char Array تحتوي على الحروف الموجودة في عمود نصي، وهي مماثلة للوسيلة السابقة في المعاملات والقيمة العائدة، فيما عدا أن المعامل الثالث يستقبل مصفوفة حروف بدلا من مصفوفة الوحدات الثنائية.

### هل القيمة منعدمة IsDBNull:

تعيد True إذا كانت الخانة التي أرسلت رقمها كمعامل فارغة DbNull.. لاحظ أن عليك استخدام هذه الوسيلة قبل محاولة قراءة قيمة أي خانة، فقارئ البيانات يسبب خطأ إذا كانت قيمة الخانة NULL.. هكذا مثلا يمكنك محاولة قراءة الخانة الموجودة في العمود الأول في الصف الحالي:

```
If Not Reader.IsDBNull(0) Then MsgBox(Reader(0))
```

كما تمتلك هذه الواجهة عددا من الوسائل التي تستقبل رقم العمود، وتعيد قيمة الخانة الموجودة في هذا العمود في الصف الحالي.. وتختلف هذه الوسائل في نوع القيمة العائدة منها، حيث تقوم كل منها بتحويل بيانات الخانة إلى أحد أنواع إطار العمل الأساسية، كما هو موضح في الجدول التالي:

| | الوسيلة | النوع الذي تعيده |
|---|---|---|
| ◆ | GetByte | وحدة ثنائية Byte. |
| ◆ | GetChar | حرف Char. |
| ◆ | GetBoolean | قيمة منطقية Boolean. |
| ◆ | GetInt16 | عدد قصير Short. |
| ◆ | GetInt32 | عدد صحيح Integer. |
| ◆ | GetInt64 | عدد طويل Long. |
| ◆ | GetFloat | عدد مفرد Single. |
| ◆ | GetDouble | عدد مزدوج Double. |
| ◆ | GetDecimal | عدد عشري Decimal. |
| ◆ | GetDateTime | تاريخ ووقت DateTime. |
| ◆ | GetString | نص String. |
| ◆ | GetGuid | سجل المعرف المتفرد Guid Structure. |

وتسبب هذه الوسائل خطأ في البرنامج إذا فشلت في تحويل البيانات إلى النوع المطلوب.

## فئة سجل البيانات DbDataRecord Class

هذه الفئة تمثل الواجهة IDataRecord، وهي تمتلك كل وسائلها وخصائصها دون أن تزيد عليها شيئا.
وتستخدم هذه الفئة مع واجهة العداد للمرور عبر سجلات قارئ البيانات، كما سنرى لاحقا.

## واجهة قارئ البيانات IDataReader Interface

ترث هذه الواجهة كلا من الواجهتين IDisposable و IDataRecord.
وإضافة إلى ما ترثه من خصائص، تمتلك هذه الواجهة الخصائص الجديدة التالية:

**العمق Depth:**
تعيد رقما يمثل عمق الجدول الحالي، إذا كانت النتيجة تحتوي على جداول متداخلة (خانات بها جداول، بها خانات بها جداول... إلخ)، مع ملاحظة أن الجدول الخارجي يكون عمقه صفرا، وأول جدولا داخلي عمقه ١ ... وهكذا.

**هل هو مغلق IsClosed:**
تعيد True إذا تم إغلاق قارئ البيانات.

**السجلات المتأثرة RecordsAffected:**
تعيد عدد الصفوف التي تأثرت بأوامر الإضافة أو التحديث أو الحذف.. وتعيد هذه الوسيلة ٠ إذا لم تتأثر أية سجلات أو فشل تنفيذ الاستعلام، وتعيد -١ إذا كان الاستعلام يستخدم الأمر SELECT.
لاحظ أن هذه الخاصية لا تعطيك القيمة الصحيحة إلا بعد إغلاق قارئ البيانات، لهذا عليك أن تتأكد أولا أن للخاصية IsClosed القيمة True، أو تستخدم الخاصة RecordsAffected بعد استخدام الوسيلة Close.

كما تمتلك هذه الواجهة هذه الوسائل الجديدة:

### قراءة Read:

تجعل قارئ البيانات ينتقل إلى السجل التالي، وتعيد True.. أما إذا كان السجل الحالي هو آخر سجل ولا يوجد سجل تال، فإنها تعيد False، وعليك التوقف عن القراء في هذه الحالة، وإلا حدث خطأ.

لاحظ أن قارئ البيانات يشير مبدئيا إلى السجل رقم -١، أي أنه يشير إلى السجل السابق لأول سجل، وهذا سيجعل محاولة القراءة تسبب خطأ في البرنامج، حيث ستخبرك رسالة الخطأ أن هذه محاولة غير مسموح بها للقراءة بينما لا توجد بيانات حاليا:

Invalid attempt to read when no data is present.

لهذا عليك استدعاء الوسيلة Read أولا للانتقال إلى أول سجل وقراءته، ثم الاستمرار في استدعائها إلى أن تعيد False، وذلك على الصيغة التالية:

```
Do While Reader.Read
    ' الكود اللازم لقراءة السجل الحالي
Loop
```

### النتيجة التالية NextResult:

عند استخدام كائن الأمر لتنفيذ أكثر من جملة SQL، أو تنفيذ إجراء مخزن يعيد أكثر من نتيجة، فإن قارئ البيانات يشير مبدئيا إلى أول نتيجة، وعليك بعد قراءة كل سجلاتها أن تستخدم هذه الوسيلة لجعل قارئ البيانات يشير إلى النتيجة التالية.. وتعيد هذه الوسيلة True إذا وجدت نتيجة تالية، لهذا عليك أن تستمر في استدعائها إلى أن تعيد False، وذلك على الصيغة التالية:

```
Do
  Do While Reader.Read
      ' الكود اللازم لقراءة السجل الحالي
  Loop
Loop While Reader.NextResult
```

## قراءة جدول المخطط GetSchemaTable:

تعيد كائن جدول DataTable فارغا يحتوي على مخطط النتيجة التي يتعامل معها قارئ البيانات.. وسنتعرف على كائن الجدول لاحقا.
لاحظ أن المخطط يحتوي على اسماء الأعمدة وأنواع بياناتها وأحجامها.. ويمكنك أن ترى هذا المخطط بشكل عملي في المشروع SchemaTable..
في هذا المشروع استخدمنا قارئ بيانات ليحمل جدول المؤلفين، واستخدمنا الوسيلة GetSchemaTable للحصول على مخطط جدول المؤلفين، وعرضناه في جدول عرض البيانات DataGridView الذي سنتعرف عليه بالتفصيل في فصل لاحق.

## إغلاق Close:

تغلق قارئ البيانات.. هذا ضروري لتحرير كائن الاتصال المرتبط بقارئ البيانات، لأنك لن تستطيع استخدام كائن الاتصال في أي عملية أخرى طالما كان قارئ البيانات يستخدمه.
وكما ذكرنا من قبل، لو أنشأت قارئ البيانات باستخدام الوسيلة ExecuteReader بالصيغة التالية (حيث Cmd هو اسم كائن الأمر):

```
Dim Dr = Cmd.ExecuteReader(
            CommandBehavior.CloseConnection)
```

فإن كائن الأمر Dr سيقوم بإغلاق الاتصال بقاعدة البيانات تلقائيا بمجرد استدعاء الوسيلة Close.

## فئة قارئ البيانات DbDataReader Class

هذه الفئة أساسية مجردة تجب وراثتها، وهي موجودة في النطاق:
System.Data.Common
وتمثل هذه الفئة الواجهة IDataReader، مما يعني أنها تمثل أيضا الواجهتين IDataRecord و IDisposable.. كما أنها تمثل أيضا واجهة القابلية للعد IEnumerable، مما يعني أنها تمتلك الوسيلة GetEnumerator التي تعيد عدادا يمر عبر سجلات النتيجة واحدا بعد الآخر، مع ملاحظة أن كل عنصر في هذا العداد هو من نوع الفئة DbDataRecord.. لهذا تستطيع المرور عبر سجلات قارئ البيانات باستخدام الوسيلة Read كما شرحنا سابقا، أو باستخدام حلقة التكرار For Each على الصيغة التالية:

```
For Each R As DbDataRecord In Dr
    MsgBox(R(0).ToString( ))
    MsgBox(R(1).ToString( ))
Next
```

هذا الكود سيعرض محتويات أول وثاني حقل في كل سجل من سجلات قارئ البيانات، بافتراض أن قارئ البيانات اسمه Dr وأن النتيجة بها حقلان أو أكثر.
وإضافة إلى ما ترثه من خصائص، تمتلك الفئة DbDataReader الخاصيتين الجديدتين التاليتين:

### به صفوف HasRows:

تعيد True إذا كان قارئ البيانات يتعامل مع نتيجة بها صفوف.

### عدد الحقول المرئية VisibleFieldCount:

تعيد عدد الأعمدة غير الخفية في قارئ البيانات.

كما تمتلك هذه الفئة الوسائل الجديدة التالية:

### معرفة نوع الحقل طبقا للمزود GetProviderSpecificFieldType:

أرسل إلى هذه الوسيلة رقم العمود، لتعيد إليك كائن النوع Type الذي يمثل نوع بياناته.. هذا النوع لن يكون من أنواع إطار العمل الأساسية، بل سيكون من الأنواع الخاصة بمزود البيانات.. على سبيل المثال، لو كنت تتعامل مع

مزود سيكويل سيرفر، وكان العمود يحتوي على نصوص، فإن هذه الوسيلة ستعيد كائنا يمثل نوع الفئة SqlString وليس الفئةString.

### قراءة القيمة طبقا للمزود GetProviderSpecificValue:

أرسل إلى هذه الوسيلة رقم العمود، لتعيد إليك كائنا Object يحمل قيمتـــه.. لاحظ أن هذه القيمة ستكون من الأنواع الخاصة بمـــزود البيانـــات، لهـــذا سيسبب المثال التالي خطأ (بافتراض أن العمود رقم صفر عمود نصي):

```
Dim Name As String = Dr.GetProviderSpecificValue(0)
MsgBox(Name)
```

والصواب أن تستخدم الكود التالي:

```
Dim Name As SqlString = Dr.GetProviderSpecificValue(0)
MsgBox(Name.Value)
```

### قراءة القيم طبقا للمزود GetProviderSpecificValues:

أرسل إلى هذه الوسيلة مصفوفة كائنات Object Array، لتملأها لك ببيانات الصف الحالي في قارئ البيانات.. وتعيد هذه الوسيلة عدد الخانات التي تـــم ملؤها في المصفوفة.

والفئات التالية ترث الفئة DbDataReader:

١- DataTableReader Class
٢- OdbcDataReader Class
٣- OleDbDataReader Class
٤- SqlDataReader Class
٥- OracleDataReader Class

وسنكتفي هنا بالتعرف على الفئة SqlDataReader، وسنتعرف في فصل لاحق على الفئة DataTableReader.

## فئة قارئ بيانات سيكويل SqlDataReader Class

هذه الفئة ترث الفئة DbDataReader بكل وسائلها وخصائصها، وهي تمكنك من قراءة البيانات القادمة من خادم سيكويل.
وليس لهذه الفئة حدث إنشاء، ولكنك تستطيع الحصول على نسخة منها باستدعاء الوسيلة ExecuteReader الخاصة بأمر سيكويل SqlCommand.
ولا تمتلك هذه الفئة أية خصائص جديدة غير ما ترثه من الفئة الأم، ولكنها تمتلك العديد من الوسائل الجديدة، وهي تقوم بقراءة البيانات من العمود الذي أرسلت رقمه إليها كمعامل، وتحولها إلى نوع البيانات المطلوب.. ولأسماء هذه الوسائل الصيغة العامة GetX، حيث X هو اسم نوع البيانات الذي سيتم التحويل إليه.. وهذه الوسائل هي:

- GetSqlBinary
- GetSqlBoolean
- GetSqlByte
- GetSqlBytes
- GetSqlChars
- GetSqlDateTime
- GetSqlDecimal
- GetSqlDouble
- GetSqlGuid
- GetSqlInt16
- GetSqlInt32
- GetSqlInt64
- GetSqlMoney
- GetSqlSingle
- GetSqlString
- GetSqlValue
- GetSqlValues
- GetSqlXml
- GetTimeSpan

على سبيل المثال، الوسيلة GetSqlBinary تعيد كائنا من النوع SqlBinary، وهي مناسبة لقراءة الأعمدة التي تحتوي بيانات من النوع image أو varbinary(MAX)، لهذا استخدماها في الزر GetSqlBinary في المشروع ReadLargeData، لقراءة صورة أول ناشر من العمود Logo2 (ويمكنك استخدامها أيضا للقراءة من العمود Logo)، وحفظها في ملف.. لاحظ أن عملية القراءة ستتم هنا بطريقة مباشرة (غير تتابعية)، وأن حجم الصورة سيؤثر على كفاءة هذه العملية، فلو كانت ضخمة فسيأخذ نقلها وقتا طويلا، وسيتم تحميلها في الذاكرة كاملة قبل حفظها في الملف!

بهذا نكون قد أكملنا تعرفنا على قارئ البيانات.. وللتدريب على ما تعلمناه حتى الآن، يمكنك فحص المشروع AuthorBooks_Reader.. في هذا المشروع نسمح للمستخدم بكتابة اسم المؤلف في مربع نص، وعندما يضغط الزر "عرض الكتب"، نستخدم قارئ البيانات لكتابة كتب هذا المؤلف في مربع نص متعدد الأسطر.

لا تنسَ نسخ قاعدة بيانات الكتب من القرص الضوئي إلى المحرك :C والتأكد من أنها ليست للقراءة فقط، لكي يعمل المثال بشكل صحيح.

| ملحوظة: |
|---|
| لاستخدام الإجراء المخزن GetAuthorBooks للحصول على كتب المؤلف الذي ترسل إلى هذا الإجراء اسمه كمعامل، أجر التعديلات التالية على المشروع AuthorBooks_Reader:<br>– ضع في الخاصيّة CommandText الخاصة بكائن الأمر اسم الإجراء المخزّن GetAuthorBooks.<br>– ضع في الخاصية CommandType الخاصة بكائن الأمر القيمة StoredProcedure.<br>– لا تُجر أية تعديلات على المعامل Author@ الذي أضفناه إلى مجموعة معاملات كائن الأمر.<br>هذا فقط هو كل المطلوب، وسيعمل البرنامج بشكل سليم، وسيعطي نفس النتائج التي كان يعطيها سابقا، مع اختلاف واحد: أنه يستخدم الإجراء المخزن بدلا من جملة SQL.<br>والمشروع AuthorBooks_Reader2 يحتوي بالفعل على هذه التعديلات. |

-١٠-

# موصل البيانات DataAdapter

موصل البيانات هو مجرد حلقة وصل بين كائن الاتصال Connection Object ومجموعة البيانات DataSet، ومهمته هي إيصال البيانات من الخادم إلى مجموعة البيانات أو العكس.

ويتكون موصل البيانات في الحقيقة من أربعة كائنات أوامر Command Objects، تحتوي على جمل SQL اللازمة لتحديد البيانات من الخادم SELECT، وتحديثها UPDATE وإدراج سجلات جديدة INSERT وحذف سجلات موجودة DELETE، وبهذا يتيح لمجموعة البيانات حرية التعامل مع قاعدة البيانات في كلا الاتجاهين (الاستقبال والإرسال)، على عكس قارئ البيانات DataReader، الذي يقرأ البيانات فقط.

في الحقيقة، فإن موصل البيانات يستخدم قارئ البيانات Data Reader داخليا لتنفيذ أمر التحديد.. لكن هذا مجرد جزء من قدرات موصل البيانات، فهو يستطيع إرسال التغييرات من مجموعة البيانات إلى قاعدة البيانات، كما يتيح لك عمل خراط للجداول Table Mapping، وذلك بإعادة تسمية الجداول والأعمدة بأسماء خاصة بك، وربطها بالأسماء الحقيقية في قاعدة البيانات، وهو ما سنتعرف عليه بالتفصيل في نهاية هذا الفصل.

دعنا إذن نتعرف على الواجهات والفئات التي تصنع موصل البيانات.

## واجهة موصل البيانات IDataAdapter Interface

تقدم هذه الواجهة العناصر الأساسية لموصل البيانات، وهي تمتلك الخصائص التالية:

### خرائط الجداول TableMappings:

تعيد كائنا يمثل المجموعة ITableMappingCollection، يحتوي على خرائط الجداول التي يستخدمها موصل البيانات لربط الجداول من مجموعة البيانات بالجداول الأصلية في قاعدة البيانات.. وسنتعرف على هذا الموضوع بالتفصيل لاحقا.

### التصرف عند غياب الخريطة MissingMappingAction:

تحدد ماذا سيحدث في حالة عدم وجود خريطة لبعض الجداول في مجموعة البيانات، وهي تأخذ إحدى قيم المرقم MissingMappingAction التي سنتعرف عليها لاحقا.
والقيمة الافتراضية لهذه الخاصية هي Passthrough، مما يعني أن مجموعة البيانات تستخدم نفس أسماء جداول قاعدة البيانات، ولا حاجة إلى وجود خريطة للجداول في هذه الحالة.

### التصرف عند غياب المخطط MissingSchemaAction:

تحدد ماذا سيحدث في حالة عدم وجود بعض الجداول أو الأعمدة في مجموعة البيانات، وهي تأخذ إحدى قيم المرقم MissingSchemaAction التي سنتعرف عليها لاحقا.
والقيمة الافتراضية لهذه الخاصية هي Add، مما يعني أن الجدول الناقص أو العمود الناقص سيضاف إلى مجموعة البيانات تلقائيا عند ملئها بالبيانات.

كما تمتلك هذه الواجهة الوسائل التالية:

### ملء Fill:

أرسل إلى هذه الوسيلة كائن مجموعة البيانات DataSet التي تريد ملأها بالجداول والسجلات الناتجة من تنفيذ أمر التحديد SELECT الخاص بموصل البيانات.. وإذا كانت مجموعة البيانات تحتوي على السجلات فعلا، فسيحدث أحد الاحتمالين التاليين:

١- إذا كان هناك مفتاح أساسي Primary Key أو قيد تفرد Unique Constraint لسجلات مجموعة البيانات، فسيتم إنعاشها من جديد بأحدث قيم موجودة في قاعدة البيانات.

٢- إذا لم يكن هناك ما يميز كل سجل ويمنع تكراره، فستضاف السجلات إلى مجموعة البيانات مرة أخرى، مما يجعلها مكررة!

وتعيد هذه الوسيلة عدد السجلات التي تمت إضافتها أو تحديثها.

لاحظ أنك لا تحتاج إلى فتح الاتصال مع قاعدة البيانات أولا، فهذه الوسيلة تقوم بفتحه إن كان مغلقا ثم تعيد إغلاقه.. أما لو كان الاتصال مفتوحا قبل استدعاء هذه الوسيلة، فإنها تستخدمه ثم تتركه مفتوحا كما هو.

ويسمى كل جدول يضاف إلى مجموعة البيانات تبعا لخريطة الجدول Table Mapping إن وجدت.. فإن لم توجد هذه الخريطة، تستخدم قواعد التسمية التالية:

١- إذا كان أمر التحديد يعيد سجلات جدول واحد فقط، فإنها تضاف في جدول يسمى Table.

٢- إذا كان أمر التحديد ينفذ أكثر من استعلام ويعيد سجلات أكثر من جدول، فإن كل نتيجة منها توضع في جدول مستقل، ويتم تسمية هذه الجداول بالترتيب Table و Table1 و Table2... وهكذا.. لاحظ أن حدوث أي خطأ في أي استعلام، سيمنع تنفيذ الاستعلامات التالية له ولن توضع باقي النتائج في مجموعة البيانات.

٣- يمكنك استخدام أكثر من موصل بيانات لملء نفس مجموعة البيانات بالجداول.. في هذه الحالة سيضيف موصل البيانات الأول جدولا اسمه Table، وسيحاول موصل البيانات الثاني أن يضيف جدولا اسمه Table أيضا، لكن نظرا لأنه موجود، فسترفضه مجموعة البيانات ولن تتم إضافته، لكن لن يحدث خطأ في البرنامج!.. هذا يوضح لك ضرورة استخدام خريطة الجدول لتسمية كل من الجدولين باسمين مختلفين لحل هذه المشكلة.

٤- يسمى كل عمود في مجموعة البيانات، بنفس اسمه في الجدول الأصلي في قاعدة البيانات.

٥- إذا كانت النتيجة تحتوي على أكثر من عمود بنفس الاسم (بسبب استخدام استعلام يجمعها من أكثر من جدول من قاعدة البيانات) فإنها توضع في الجدول بعد إضافة الأرقام (١، ٢، ٣ ... ) إلى نهاية اسم العمود لمنع التشابه.

٦- إذا كانت بعض الأعمدة بدون أسماء (لأنها ناتجة عن دوال تجميع مثلا) فإنها تعطى الأسماء الافتراضية Column1 و Column2 و Column3... وهكذا.

ويجب عليك ألا تضيف إلى مجموعة البيانات جداول أو أعمدة خاصة بك وتسميها بهذه الأسماء الافتراضية، كي لا يحدث أي تعارض أو خطأ بسببها. وستجد مثالا على استخدام الوسيلة Fill في المشروع DataGridViewAuthorBooks.. في هذا المشروع نستخدم موصل بيانات اسمه DAAuthors لتحميل سجلات المؤلفين من قاعدة البيانات، كما نستخدم موصل بيانات اسمه DABooks لتحميل سجلات الكتب من قاعدة البيانات.. ولملء مجموعة بيانات اسمها Ds بسجلات المؤلفين والكتب، نستخدم الكود التالي في حدث تحميل النموذج Form1_Load:

```
DAAuthors.Fill(Ds)
DABooks.Fill(Ds)
```

### ملء المخطط FillSchema:

تملأ مجموعة البيانات بمخطط البيانات Schema.. هذا معناه أن الجداول الناتجة عن الاستعلام وتفاصيل أعمدتها ستضاف إلى مجموعة البيانات، لكن دون إضافة أي سجلات إليها.. بتعبير آخر: سيتم ملء مجموعة البيانات بجداول فارغة.

وتستقبل هذه الوسيلة معاملين:

- كائن مجموعة البيانات DataSet الذي سيتم ملؤه بالمخطط.
- إحدى قيم المرقم SchemaType، تحدد ماذا سيحدث لو كان موصل البيانات يحتوي مسبقا على خرائط للجداول والأعمدة Mappings، وهذه القيم هي:

| | |
|---|---|
| Source | تجاهل خرائط الجداول وخرائط الأعمدة، وملء مجموعة البيانات بنفس أسماء الجداول والأعمدة الأصلية الموجودة في المخطط Schema. |
| Mapped | استخدام خرائط الجداول وخرائط الأعمدة، وملء مجموعة البيانات بالأسماء الموجودة في هذه الخرائط بدلا من أسماء الجداول والأعمدة الأصلية. |

ويتضمن المخطط الذي يتم ملء مجموعة البيانات به التفاصيل التالية:

- اسم الجدول، وأسماء الأعمدة.
- خصائص كل عمود، مثل:

أ. السماح بتركه فارغا AllowDBNull.

ب. هل هو متفرد Unique.

ج. هل هو للقراءة فقط ReadOnly.

د. هل يزيد تلقائيا AutoIncrement.. لكن عليك أنت تحديد معدل الزيادة وبداية العداد، فهما لا يضافان تلقائيا.

هـ. أقصى طول للبيانات في العمود MaxLength.

- إذا كان المفتاح الأساسي Primary Key موجودا ضمن أعمدة النتيجة، يتم استخدامه كمفتاح أساسي للجدول في مجموعة البيانات.. وإذا لم يوجد مفتاح أساسي وكان هناك حقل متفرد القيمة Unique، يتم استخدامه كمفتاح أساسي للجدول في مجموعة البيانات، بشرط ألا يكون مسموحا بتركه فارغا (AllowDbNull = False).. أما إذا كان الحقل المتفرد يقبل القيمة NULL، فلن يستخدم كمفتاح أساسي، وستكتفي هذه الوسيلة بإضافة قيد التفرد UniqueConstraint الخاص بهذا العمود إلى مجموعة القيود ConstrainsCollection الخاصة بالجدول.
- أي قيود أخرى غير المفتاح الأساسي وقيد التفرد لا تضاف إلى الجدول، وعليك إضافتها بنفسك!

وتعيد هذه الوسيلة مصفوفة جداول DataTable Array، تحتوي على كائنات الجداول التي تمت إضافة مخططاتها إلى مجموعة البيانات.

ويمكنك بعد استخدام هذه الوسيلة، استخدام الوسيلة Fill لملء الجداول بالبيانات.. لكن كما ذكرنا سابقا، فإن استدعاء الوسيلة Fill بمفردها يضيف مخططات الجداول والسجلات معا إلى مجموعة البيانات، مما يغني في معظم الحالات عن استدعاء الوسيلة FillSchema أولا.

إذن.. فما فائدة هذه الوسيلة؟

تفيدك هذه الوسيلة إذا أردت عرض الجداول فارغة للمستخدم ليدخل بيانات جديدة دون أن يعبث بالبيانات القديمة.. هذا هو ما فعلناه في المشروع UpdateErrors، حيث سيعرض جدول البيانات أعمدة جدول المؤلفين،

لكنه لن يعرض بيانات أي مؤلفين، وبهذا يستطيع المستخدمن إدخال مؤلفين جددا وحفظهم في قاعدة البيانات، دون أن يغير بيانات المؤلفين السابقين.
تفيدك أيضا إذا أردت إنشاء المفتاح الأساسي في جدول مجموعة البيانات، فاستخدام الوسيلة Fill بمفردها لا ينشئ المفتاح الأساسي في الجدول الذي تضيفه إلى مجموعة البيانات، وهذا قد يسبب لك مشاكل في بعض الحالات التي تحتاج فيها إلى المفتاح الأساسي، كما يحدث عند استخدام الوسيلة DataSet.Merge لدمج السجلات، أو عند البحث عن سجل بدلالة مفتاحه الأساسي... إلخ.

**معرفة معاملات الملء GetFillParameters:**
تعيد مصفوفة معاملات IDataParameter Array، تحتوي على المعاملات التي استخدمها كائن الأمر عند تنفيذ أمر التحديد SELECT الخاص بموصل البيانات.

**تحديث Update:**
أرسل إلى هذه الوسيلة كائن مجموعة البيانات DataSet التي تريد نقل التغييرات التي حدثت على سجلاتها إلى قاعدة البيانات.. وتعيد هذه الوسيلة عدد السجلات التي نجح تحديثها في قاعدة البيانات.
وتستخدم هذه الوسيلة أوامر الإضافة والحذف والتحديث الخاصة بموصل البيانات، لنقل التغييرات من مجموعة البيانات إلى قاعدة البيانات، تبعا للتغيير الذي حدث لكل سجل.. ويحدث خطأ في البرنامج، إذا لم يوفر موصل البيانات الأمر المطلوب من هذه الأوامر.
لاحظ أن هذه الوسيلة ذكية، فهي تمر عبر كل سجل في مجموعة البيانات، وترى إن كان هناك أي تغيير قد حدث لهذا السجل، ومن ثم تستخدم الأمر المناسب لإرسال هذا التغيير إلى مجموعة البيانات.. أما السجلات التي لم يحدث بها أي تغيير، فيتم تجاهلها.. وبهذا لا تضيع هذه لوسيلة أي وقت في محاولة حفظ سجلات لم يحدث فيها تغيير.. لذا فأنت لا تحتاج إلى القيام بأية خطوات خاصة لتحسين وظيفة هذه الوسيلة.
لكن، لو كانت مجموعة البيانات تحتوي على أكثر من جدول، فأيها يا ترى سيتم تحديثه؟

يحدد هذا موصل البيانات الذي تستخدمه.. مثلا: لو استخدمت موصل بيانات لملء مجموعة البيانات بجدول المؤلفين، فإن الوسيلة Update الخاصة به ستتعامل مع سجلات جدول المؤلفين.. ولو استخدمت موصل بيانات لملء

مجموعة البيانات بجدول الكتب، فإن الوسيلة Update الخاصة به ستتعامل مع سجلات جدول الكتب.. لهذا تحتاج الجملتين التاليتين لحفظ التغييرات:

```
DaAuthors.Update(Ds)
DaBooks.Update(Ds)
```

لاحظ أن هذا الكود صحيح ولكنه قد يسبب مشاكل في بعض الحالات، والأفضل أن تحاول تحديث الجدول الفرعي قبل الجدول الرئيسي، فلو كنت حذفت مؤلفا وكتبه، فإن محاولة تحديث جدول المؤلفين أولا ستحاول حذف سجل هذا المؤلف، وهذا سيسبب خطأ إذا كنت فرضت قيد المفتاح الفرعي Foreign Key Constraint، لأن كتب هذا المؤلف ستشير إلى مؤلف غير موجود.. بينما لو عكست العملية، وحدثت جدول الكتب أولا، فسيتم حذف كتب المؤلف بلا مشاكل، ومن ثم يتم حذف المؤلف نفسه عند تحديث جدول المؤلفين.. لهذا عليك استخدام الكود التالي في عملية التحديث:

```
DaBooks.Update(Ds)
DaAuthors.Update(Ds)
```

وهذا هو الكود الذي استخدمناه في الزر "حفظ في قاعدة البيانات" في المشروع DataSetSample.

## واجهة موصل بيانات قاعدة البيانات
## IDbDataAdapter Interface

هذه الواجهة ترث الواجهة IDataAdapter، وهي تمد موصل البيانات بالأوامر اللازمة للتعامل مع قاعدة البيانات.
وإضافة إلى ما ترثه من الواجهة الأم، تمتلك هذه الواجهة الخصائص التالية:

### أمر التحديد SelectCommand:

هذه الخاصية من نوع الواجهة IDbCommand، وهي تستقبل كائن الأمر الذي يحتوي على جملة التحديد SELECT التي سيستخدمها موصل البيانات للحصول على السجلات من قاعدة البيانات عند استدعائك للوسيلة Fill أو FillSchema.. ويمكن أن يحتوي أمر التحديد على مجموعة من الأوامر SQL Batch، وفي هذه الحالة يضيف موصل البيانات أكثر من جدول إلى مجموعة البيانات.

### أمر الإدراج InsertCommand:

هذه الخاصية من نوع الواجهة IDbCommand، وهي تستقبل كائن الأمر الذي يحتوي على جملة الإدراج INSERT التي يستخدمها موصل البيانات عند استدعاء الوسيلة Update، لإضافة السجلات الجديدة في مجموعة البيانات إلى قاعدة البيانات.
هذا المشروع يستخدم استعلام الإدراج التالي:

**INSERT INTO Authors**
**(Author, CountryID, Phone, About)**
**VALUES (@Author, @CountryID, @Phone, @About)**

ويريك المشروع CopyAuthors كيف يمكن تعريف أمر الإدراج ومعاملاته، ووضعه في الخاصية InsertCommand، واستدعاء الوسيلة Update لاستخدامه في حفظ السجلات الجديدة من مجموعة البيانات إلى قاعدة البيانات.. لاحظ أن معاملات أمر الإدراج مربوطة بمجموعة البيانات، لأخذ قيمها من أعمدتها مباشرة.. لقد عرفنا من قبل أن كائن المعامل Parameter Object يملك الخاصية SourceColumn التي نضع فيها اسم العمود الذي سنقرأ القيمة منه من مجموعة البيانات.. وستجد أننا أرسلنا القيمة إلى هذه الخاصية من خلال المعامل الرابع لحدث الإنشاء New عند تعريف كل معامل، كما هو موضح في الكود التالي:

```
InsertCmd.Parameters.AddRange(New SqlParameter( ) {
        New SqlParameter("@Author",
            SqlDbType.NVarChar, 0, "Author"),
        New SqlParameter("@CountryID",
            SqlDbType.SmallInt, 0, "CountryID"),
        New SqlParameter("@Phone",
            SqlDbType.VarChar, 0, "Phone"),
        New SqlParameter("@About",
            SqlDbType.NVarChar, 0, "About")})
```

لهذا لا نحتاج إلى كتابة أي كود لقراءة القيم من مجموعة البيانات، فعند استدعاء الوسيلة Update، فإنها تمر عبر كل صف من صفوف مجموعة البيانات، وتأخذ قيم أعمدته وتمررها إلى معاملات أمر الإدراج، وتنفذ أمر الإدراج.

| ملحوظة: |
| --- |
| لإنعاش السجل المعروض في مجموعة البيانات، يمكنك استخدام جملة SELECT بعد استعلام الإدراج أو التحديث الخاص بقواعد بيانات سيكويل سيرفر، وذلك بوضع فاصلة منقوطة ; بين الأمرين.. مثلا:<br>INSERT INTO Authors<br>(Author, CountryID, Phone, About)<br>VALUES (@Author, @CountryID, @Phone, @About);<br>SELECT * FROM Authors<br>WHERE ID = SCOPE_IDENTITY( )<br>هذا مفيد في بعض الحالات.. مثلا: لو كان السجل يحتوي على عمود ترقيم تلقائي، فإن الرقم الذي ستعطيه مجموعة البيانات للسجل هو مجرد اقتراح لا تعمل قاعدة البيانات به (لهذا ليست مشكلة أن تضع فيه مجموعة البيانات ٠ أو -١)، حيث تعطي قاعدة البيانات للسجل الرقم الصحيح في الترقيم عند إضافته إليها، لهذا تكون جملة SELECT مفيدة لعرض الترقيم الصحيح للسجل في برنامجك.<br>أما إذا لم تجد داعيا لإنعاش مجموعة البيانات، فلا تستخدم أمر التحديد.<br>لاحظ أن الدالة SCOPE_IDENTITY تعيد آخر معرف تم توليده في نطاق التنفيذ الحالي، وهو بالطبع معرف السجل الذي أضفناه للتو.. ولا ينصح باستخدام الدالة @@IDENTITY لأنها قد تتأثر بعوامل أخرى في قاعدة البيانات مثل المنبهات Triggers، مما يجعلها تعيد معرفا غير المعرف الخاص بالسجل الذي أضفته. |

### أمر الحذف DeleteCommand:

هذه الخاصية من نوع الواجهة IDbCommand، وهي تستقبل كائن الأمر الذي يحتوي على جملة الحذف DELETE التي يستخدمها موصل البيانات عند استدعاء الوسيلة Update، لحذف السجلات من قاعدة البيانات إذا كانت قد حذفت من مجموعة البيانات.

### أمر التحديث UpdateCommand:

هذه الخاصية من نوع الواجهة IDbCommand، وهي تستقبل كائن الأمر الذي يحتوي على جملة التحديث UPDATE التي يستخدمها موصل البيانات عند استدعاء الوسيلة Update، لنقل التغييرات من مجموعة البيانات إلى قاعدة البيانات.

| ملحوظة: |
| --- |
| إذا كانت مجموعة البيانات تحتوي على مفتاح أساسي، وكنت قد وضعت أمر التحديد في الخاصية SelectCommand، فلست مجبرا في هذه الحالة على إنشاء أوامر الحذف والإدراج والتحديث بنفسك، فموصل البيانات يستطيع إنتاج هذه الأوامر آليا عند استدعاء الوسيلة Update الخاصة به، وهو يستخدم لفعل هذا فئات بناء الأوامر CommandBuilders التي سنتعرف عليها لاحقا في هذا الفصل. |

## فئة موصل البيانات DataAdapter Class

هذه الفئة تمثل الواجهة IDataAdapter، كما أنها ترث الفئة Component.
ولا يمكنك تعريف نسخة جديدة من هذه الفئة، لأن حدث إنشائها Constructor محمي Protected، لكن يمكنك أن تتعامل مع الفئات الفرعية المشتقة منها.

وإضافة إلى ما تمثله من خصائص الواجهة IDataAdapter، تمتلك هذه الفئة الخصائص التالية:

### قبول التغييرات أثناء الملء AcceptChangesDuringFill:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتم استدعاء الوسيلة DataRow.AcceptChanges بعد إضافة كل صف إلى مجموعة البيانات، وبهذا ستعتبر مجموعة البيانات أن السجل الذي تمت إضافته لم يحدث به أي تغيير عن السجل الموجود في قاعدة البيانات.. أما لو جعلت قيمة هذه الخاصية False، فستعتبر مجموعة البيانات أن السجل المضاف إليها هو سجل جديد لم يدرج بعد في قاعدة البيانات، وعند إجراء عملية التحديث سيحاول موصل البيانات إضافته إلى قاعدة البيانات كسجل جديد، وهو أمر غير مرغوب فيه بالطبع، إلا في بعض الحالات الخاصة، كأن تقوم بتحميل السجلات من أحد الجداول، وتريد حفظها في جدول مؤقت في نفس قاعدة البيانا، أو جدول آخر في قاعدة بيانات أخرى.. لاحظ أنك في هذه الحالة ستفعل ما يلي:

١- تجعل للخاصية AcceptChangesDuringFill لموصل البيانات الأول القيمة False.
٢- نستخدم الوسيلة Fill الخاصة بموصل البيانات الأول لملء مجموعة البيانات بالسجلات.
٣- تستخدم موصل بيانات آخر لتحديث الجدول الجديد باستدعاء الوسيلة Update، مع استخدام أوامر التحديث والإدراج والحذف المناسبة للتعامل مع هذا الجدول الجديد.

والمشروع CopyAuthors يستخدم هذه الطريقة لنسخ المؤلفين من قاعدة بينات آكسيس وإضافتهم إلى قاعدة بيانات سيكويل سيرفر.. لاحظ أن الوسيلة Update لن تحتاج إلى أوامر التحديد والحذف والتحديث في حالتنا هذه، لهذا لن نعرفها، ولن يحدث خطأ في البرنامج.

### قبول التغييرات أثناء التحديث AcceptChangesDuringUpdate:

إذا جعلت قيمة هذه الخاصية True (وهـي القيمـة الافتراضـية)، فـسيتم استدعاء الوسيلة DataRow.AcceptChanges بعد نقل التغييرات من كل صف في مجموعة البيانات إلى قاعدة البيانات، وبهذا تعتبـر مجموعـة البيانات هذا الصف صفا عاديا لم يحدث به أي تغيير عن السجل الموجـود في قاعدة البيانات.. أما لو جعلت قيمـة هـذه الخاصـية False، فـستظل مجموعة البيانات تعتبر هذا الصف مختلفا عن الصف الأصلي فـي قاعـدة البيانات، وعند إجراء عملية التحديث مرة أخرى سيعاد تحديثه فـي قاعـدة البيانات، وهو أمر غير مرغوب فيه في معظم الحالات، إلا إذا كنت تريـد استخدام نفس مجموعة البيانات لتحديث أكثر من جدول.. مـثلا: إذا أردت تحديث جدول المؤلفين في كل من قاعدة بيانات آكـسيس وقاعـدة بيانـات سيكويل سيرفر، فيجب عليك فعل ما يلي:

- وضـــع القيمـــة False فـــي الخاصـــية AcceptChangesDuringUpdate لموصـل بيانـات آكـسيس واستدعاء الوسيلة Update.
- وضـــع القيمـــة True فـــي الخاصـــية AcceptChangesDuringUpdate لموصـل بيانـات سـيكويل سيرفر واستدعاء الوسيلة Update.. هذا سيجعل مجموعة البيانـات تقبل التغييرات، وتعتبر أن كل سجلاتها مطابقة للسجلات الاصـلية، فقد استخدمنا هذه التغييرات فعلا ولم نعد نحتاجها.

والمشروع UpdateAll يريك هذه الطريقة عمليا، فهـو يحمـل سـجلات المؤلفين من قاعدة بيانات سيكويل سيرفر، ويملأ بها مجموعـة البيانـات، ويعرضها للمستخدم في جدول، حيث يستطيع المستخدم إجراء التغيرات التي يريدها على بيانات المؤلفين، أو يضيف أو يحذف بعض المؤلفين، وعنـدما يضغط زر التحديث، يتم حفظ هذه التغييرات في قاعدتي آكسيس وسـيكويل سيرفر معا، وبهذا نضمن أن يظلا متزامنتين دائما.

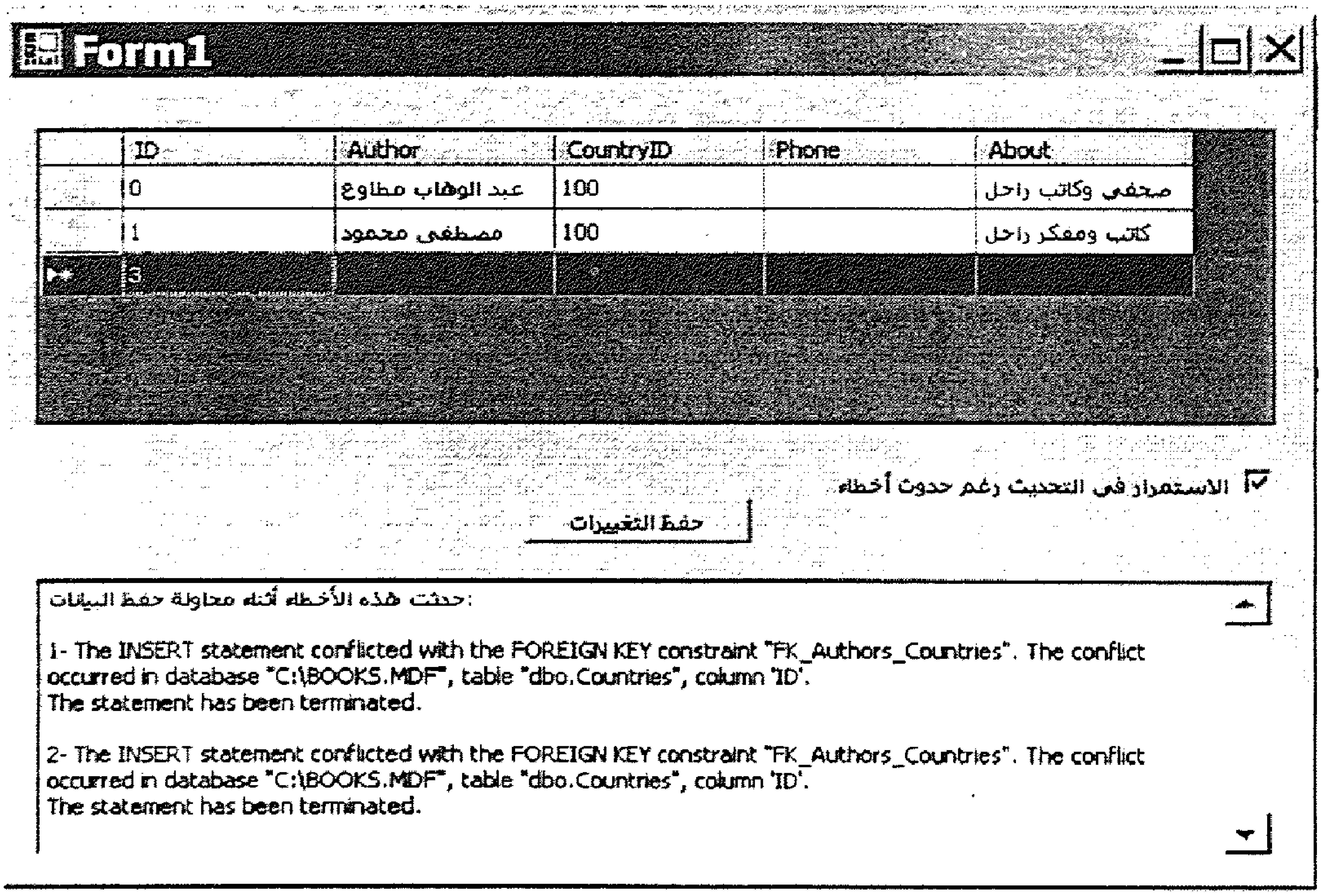
لاحظ أننا في هذا المشروع استخدمنا المعالج الـسحري Wizard لإنتـاج موصل البيانات وأوامره، لهذا لن تجد الكثير من الكـود فـي المـشروع.. وسنتعرف على هذا المعالج بعد قليل.. ومن المهم أن تلاحظ أن أمر التحديث UPDATE يتعرف على السجل المراد تحديثه في قاعدة البيانات من خلال مفتاحه الأساسي (الحقل ID في مثالنا هذا) وقيمه الأصـلية كمـا شـرحنا سابقا.. لهذا لو يكن جدول المؤلفين في قاعدة آكسيس يحتوي علـى بعـض المؤلفين الموجودين في قاعدة بيانات سيكويل سيرفر فلن يحدث خطأ، لكـن

لن يتم تحديث هؤلاء المؤلفين لأنهم غير موجودين أصلا، ولن تتم إضافتهم أيضا.. هذه المشكلة لن تحدث لو أضفت مؤلفين جددا إلى جدول العـــرض، ففي هذه الحالة سيتم حفظهم في قاعدتي البيانات بشكل صحيح.. لهـــذا لـــو أردت أن تضمن أن يعمل هذا البرنامج بصورة دقيقـــة، فيجـــب أن تجعـــل جدول المؤلفين في كلتا القاعدتين متماثلين منذ البدايـــة، ليحـــافظ عليهمـــا البرنامج هكذا باستمرار.

### استمرار التحديث عند الخطأ ContinueUpdateOnError:

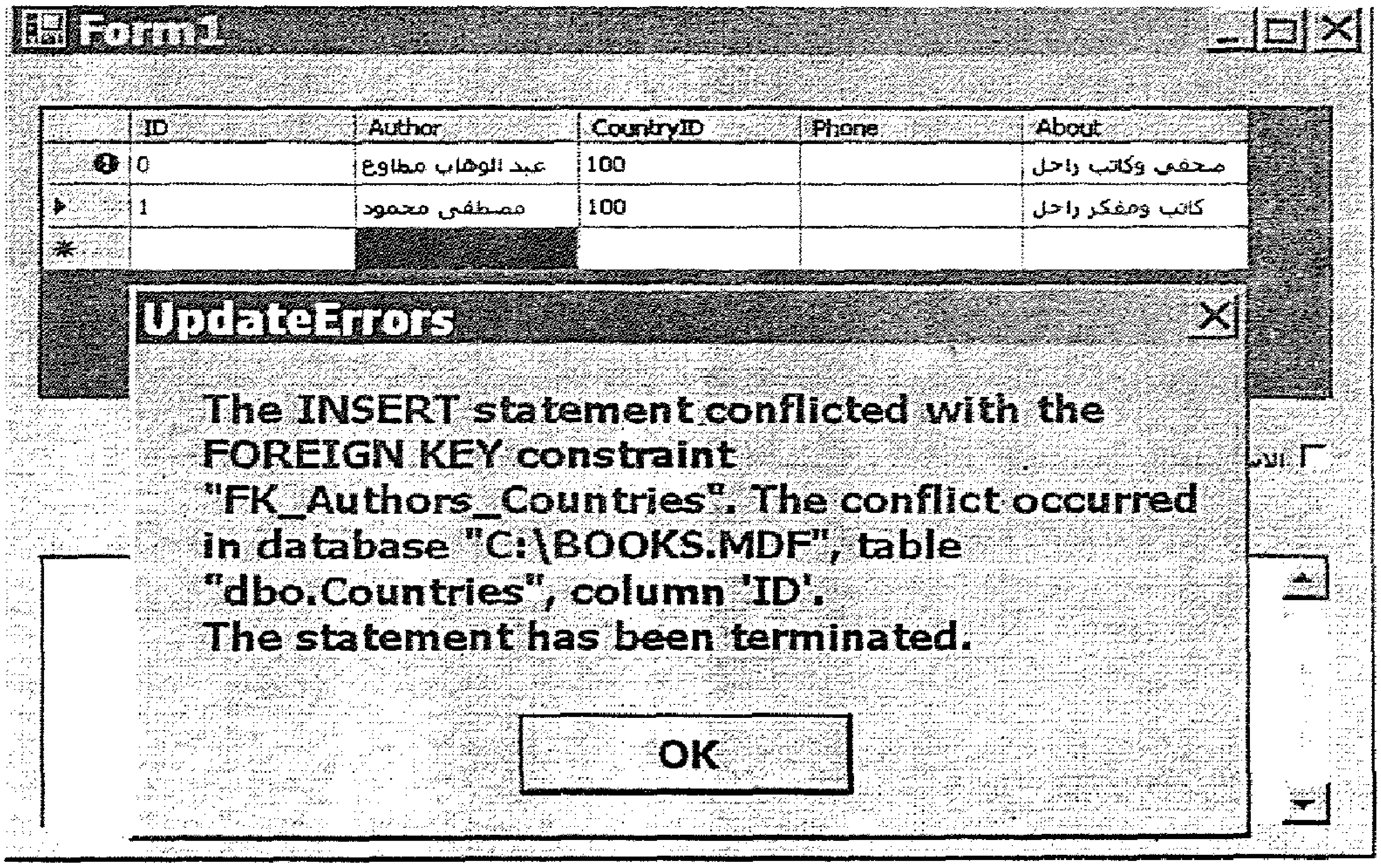
إذا جعلت قيمة هذه الخاصية True، فلن يحدث خطأ في البرنامج إذا حدثت مشكلة في تحديث أحد سجلات قاعدة البيانات، وستستمر محاولة تحديث باقي السجلات، مع إرسال تفاصيل المشكلة إلى الخاصية RowError الخاصـــة بكائن الصف DataRow الذي فشلت عملية تحديثه.

والقيمة الافتراضية لهذه الخاصية هي False، لهـــذا ســـيحدث خطـــأ فـــي البرنامج لو فشل تحديث أحد السجلات، مما سينهي عملية تحديث السجلات في الحال.. وأنت حر في اختيار الطريقة التي تناسبك أكثر لمعالجة مثل هذه الأخطاء.. والمشروع UpdateErrors يتيح للمستخدم اختيار الطريقة التي تناسبه من خلال مربع الاختيار Check Box الموضوع على النموذج.. فلو اختار "مواصلة التحديث رغم حدوث أخطاء"، فسنعرض له تقريرا بالأخطاء التي حدثت ونطلب منه إصلاحها كما توضح الصورة.



لاحظ أن جدول العرض يمنع معظم الأخطاء، فهو يرفض ترك خانة فارغة إذا كانت لا تقبل القيمة Null، كما يرفض أي نص يتكون من عدد من الحروف أطول مما تقبله الخانة.. لهذا لو أردت اختبار هذا البرنامج، فليس أمامك إلا أن تضع في العمود CountryID رقما أكبر من ٢٢، لأن قيد المفتاح الفرعي FOREIGN KEY constraint بين جدول الدول وجدول المؤلفين، يمنعك من استخدام رقم دولة ليس موجودا في جدول الدول.

أما إن اختار المستخدم إيقاف التحديث عند حدوث أخطاء، فسنستخدم المقطع Try Catch لنعرض له رسالة خطأ عند فشل تحديث أي سجل، ونحدد له هذا السجل في أدوات العرض ليقوم بتصحيحه وإعادة المحاولة.



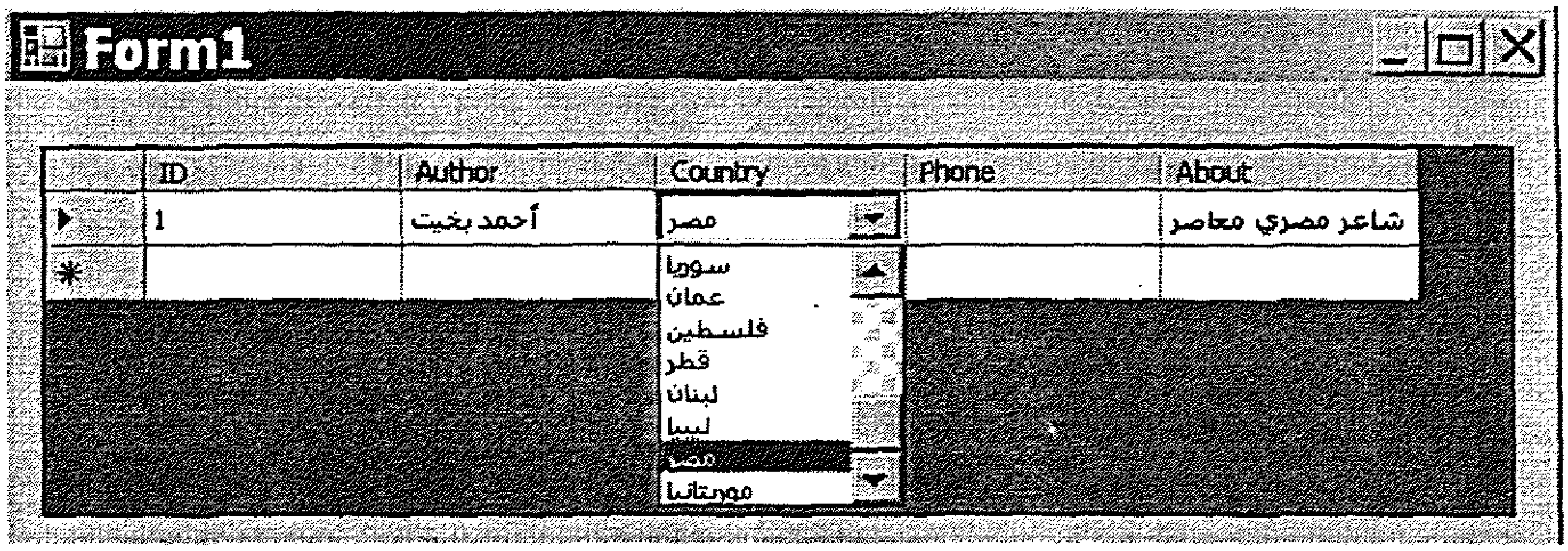

لاحظ أن السجلات التالية لهذا السجل لم تحفظ حتى الآن، وعند تكرار محاولة الحفظ سيتم حفظها، وقد تظهر أخطاء جديدة في أي سجل منها في تلك اللحظة، حيث يتوجب على المستخدم تصحيحها أيضا وإعادة المحاولة.... وهكذا.. ورغم أن هذا يبدو مملا، إلا أنه أسهل من كتابة تقرير طويل وترك المستخدم يبحث عن السجلات المذكورة في هذا التقرير لتصحيحها.

هذا هو كود زر الحفظ، مع استثناء الكود الذي يجمع أخطاء السجلات في الحالة الأولى، لأننا سنتعرف عليه في فصل لاحق:

```
If ChkContinue.Checked Then
   ' استمرار التحديث رغم حدوث أخطاء
   DaAuthors.ContinueUpdateOnError = True
   DaAuthors.Update(Ds)
   ' علينا جمع الأخطاء التي حدثت وعرضها في تقرير للمستخدم
   ' ..............................
Else ' إيقاف التحديث عند حدوث أي خطأ
   DaAuthors.ContinueUpdateOnError = False
   Try
      DaAuthors.Update(Ds)
   Catch ex As SqlException
      MsgBox(ex.Message)
   End Try
End If
```

لاحظ أننا لم نكتب أي كود لوضع علامة الخطأ التي تظهر في جدول العرض بجوار السجل الذي سبب المشكلة، فهي تظهر تلقائيا بسبب وجود ربط Binding بين جدول العرض ومجموعة البيانات.. وسنتعرف على تقنية ربط البيانات بالتفصيل في فصل لاحق.

ولعلك تلجأ في برنامجك إلى تعريب أهم رسائل الخطأ التي تتوقع حدوثها، لتعرض للمستخدم العربي رسائل يستطيع فهمها، وكذلك لمنع المستخدم من معرفة أسماء الجداول والأعمدة الموجودة في قاعدة البيانات، والتي قد يستخدمها أي قرصان لمحاولة حقن استعلامات غير مرغوبة وتخريب قاعدة البيانات.. في المثال السابق مثلا، كل ما يهم المستخدم أن يعرفه هو: "لا توجد دولة لها الرقم الذي أدخلته".. طبعا سيحتاج منك الأمر إلى بعض الجهد لكتابة كود يحلل نص الرسالة ويبني النص العربي بناء على أسماء الجداول والأعمدة الموجودة فيها.. لهذا فمن الأذكى أن تقلل احتمالات الخطأ في برنامجك إلى أقصى حد.. ففي البرامج الحقيقية، ليس على المستخدم أن يكتب أرقام الدول، بل عليك أن تعرض له قائمة منسدلة فيها أسماء الدول، وهو يختار منها مباشرة، كما هو موضح في الصورة.. هذا لا يسهل عليه الحياة فقط، بل يمنعه من إحداث أخطاء لا لزوم لها في البرنامج.. وستجد مثالا على هذا في المشروع UpdateErrors2، والذي سيتعذر عليك فيه رؤية أي مثال على أخطاء التحديث.. في هذا المشروع أيضا تخلصنا من رسالة الخطأ العقيم التي يعرضها جدول العرض عند ترك صف فيه أخطاء، وعرضنا رسالة خطأ خاصة بنا باستخدام الحدث DataGridView.DataError، مما يمنع المستخدم من معرفة أسماء الجداول والأعمدة الموجودة في قاعدة البيانات.




### خيار التحميل FillLoadOption:

تحدد هذه الخاصية ماذا سيحدث للنسخة الاصلية Original Version والنسخة الحالية Current Version من السجل عند استخدام الوسيلة Fill لملء مجموعة البيانات.

وتأخذ هذه الخاصية إحدى قيم المرقم LoadOption التالية:

| | |
|---|---|
| OverwriteChanges | تجاهل التغييرات التي حدثت سابقا للسجل، ووضع القيم القادمة من قارئ البيانات في كل من النسخة الأصلية والنسخة الحالية للسجل. |
| PreserveChanges | هذه هي القيمة الافتراضية، وهي تحافظ على النسخة الحالية للسجل بدون تغيير، ووضع القيم القادمة من قارئ البيانات في نسخة السجل الأصلية فقط. |
| Upsert | الاحتفاظ بالنسخة الأصلية للسجل بدون تغيير، ووضع القيم القادمة من قارئ البيانات في نسخة السجل الحالية فقط. |

### إعادة الأنواع الخاصة بالمزود ReturnProviderSpecificTypes:

إذا جعلت قيمة هذه الخاصية True، فستقوم الوسيلة Fill، باستخدام أنواع البيانات الخاصة بكل مزود (مثل أنواع سيكويل).. والقيمة الافتراضية لهذه الخاصية هي False، مما يجعل الوسيلة Fill تحول البيانات إلى أنواع البيانات العادية المستخدمة في إطار العمل.

كما تمتلك هذه الفئة الوسائل التالية:

**تصفير خيار التحميل ResetFillLoadOption:**

تعيد قيمة الخاصية FillLoadOption إلى قيمتها الافتراضية، وتجبر الوسيلة Fill على مراعاة قيمة الخاصية AcceptChangesDuringFill.

**حفظ خاصية قبول التغييرات أثناء الملء ShouldSerializeAcceptChangesDuringFill:**

إذا جعلت قيمة هذه الخاصية True، فسيتم الاحتفاظ بقيمة الخاصية AcceptChangesDuringFill.

**حفظ خاصية خيار التحميل ShouldSerializeFillLoadOption:**

إذا جعلت قيمة هذه الخاصية True، فسيتم الاحتفاظ بقيمة الخاصية FillLoadOption.

وتمتلك الفئة DataAdapter الحدث الوحيد التالي:

**خطأ الملء FillError:**

ينطلق إذا حدث خطأ أثناء ملء مجموعة البيانات.. والمعامل الثاني e لهذا الحدث من النوع FillErrorEventArgs، وله الخصائص التالية:

| | | |
|---|---|---|
| | DataTable | تعيد كائن جدول البيانات DataTable الذي حدث الخطأ أثناء ملئه. |
| | Errors | تعيد كائن الاستثناء Exception الذي يحتوي على معلومات الخطأ الذي حدث. |
| | Values | تعيد مصفوفة كائنات Object Array، تحتوي على القيم الموجودة بالصف الذي حدث به الخطأ. |
| | Continue | إذا جعلت قيمتها True، فسيستمر ملء الجدول بالسجلات وتجاهل الخطأ.. أما إن جعلت قيمتها False فسيتوقف ملء مجموعة البيانات في الحال. |

## فئة موصل بيانات قاعدة البيانات DbDataAdapter Class

هذه الفئة أساسية مجردة تجب وراثتها، وهي ترث الفئة DataAdapter.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخاصيتين التاليتين:

### الاسم الافتراضي لجدول المصدر DefaultSourceTableName:

قيمة هذا الثابت هي Table، وهو الاسم الافتراضي الذي يستخدم عند إضافة جدول إلى مجموعة البيانات.

### حجم مجموعة أوامر التحديث UpdateBatchSize:

ضع في هذه الخاصية عدد السجلات التي سيقوم موصل البيانات بحفظ تغييراتها في قاعدة البيانات في المرة الواحدة.. هذا مفيد لتقليل عدد دورات الاتصال مع الخادم Round Trips أثناء تحديث قاعدة البيانات، مما يجعل أداء البرنامج أفضل.. لكن عليك أن تراعي أن التعامل مع عدد كبير من السجلات في نفس اللحظة قد يؤدي إلى تقليل الكفاءة (استهلاك ذاكرة أكبر، إرسال بيانات أكثر ووقت انتظار أطول)، لهذا عليك اختيار عدد معقول من السجلات يحقق أفضل أداء.

ولا يمكنك استخدام هذه الخاصية مع آكسيس، لأنه لا يسمح بتنفيذ أكثر من أمر في المرة الواحدة.. لكن يمكنك استخدامها مع سكيويل سيرفر وأوراكل.
والجدول التالي يلخص تأثير القيم المختلفة لهذه الخاصية:

| | |
|---|---|
| ١ | تحديث سجل واحد في كل مرة.. هذه هي القيمة الافتراضية. |
| > ١ | تحديث العدد المحدد من السجلات، حيث يتم تكوين استعلام لتحديث كل سجل، ودمج هذه الاستعلامات معا بوضع ; بينها. |
| ٠ | لا توجد قيود على عدد الأوامر، وهذا معناه استخدام أكبر عدد من الأوامر يمكن لقاعدة البيانات التعامل معه. |
| < ١ | سيحدث خطأ في البرنامج! |

وفي حالة استخدام أي قيمة غير الواحد لهذه الخاصية، يجب عليك أن تضع في الخاصية UpdatedRowSource الخاصة بكائن الأمر المستخدم في تنفيذ عملية التحديث القيمة None أو OutputParameters، وإلا حدث خطأ في البرنامج.

كما أن هذه الفئة تضيف عدة صيغ جديدة لكل من الوسيلتين Fill و FillSchema.. دعنا نتعرف على هذه الصيغ:

### ملء Fill:

تضيف هذه الفئة ثلاث صيغ جديدة إلى هذه الوسيلة، وهي:

١- الصيغة الأولى تستقبل جدول بيانات DataTable لتملأه بالسجلات.. وقد استخدمنا هذه الصيغة في الوسيلة MyDbConnector.GetTable في المشروع DbTasks لملء كائن جدول DataTable بالبيانات وإعادته للمستخدم.. بعد هذا يمكنك إضافة هذا الجدول إلى مجموعة بيانات، أو عرض بياناته مباشرة في جدول عرض، أو تنفيذ أي عملية تريدها عليه.. وستجد في نفس المشروع مثالا على استخدام الوسيلة GetTable، وذلك بضغط الزر "عرض المؤلفين"، الذي يعرض نموذجا جديدا عليه جدول فيه بيانات المؤلفين.

٢- الصيغة الثانية تستقبل مجموعة البيانات المراد ملئها، ونصا يمثل اسم الجدول المضاف إلى مجموعة البيانات.. الكود التالي مثلا سيضيف جدول الكتب إلى مجموعة البيانات بالاسم TblBooks:

```
DaBooks.Fill(Ds, "TblBooks")
MsgBox(Ds.Tables(0).TableName) 'TblBooks
```

٣- الصيغة الثالثة لها أربعة معاملات، هي بالترتيب:

- مجموعة البيانات.
- رقم السجل الذي تريد القراءة بدءا منها، علما بأن أول سجل في الجدول رقمه صفر.
- أقصى عدد من السجلات تريد قراءته من الجدول.. ولن يحدث خطأ إذا كان الجدول يحتوي على عدد من السجلات أقل من هذا العدد.
- نص يمثل اسم الجدول في مجموعة البيانات.

٤- الصيغة الرابعة تفيدك عندما تريد أخذ جزء من السجلات من أمر تحديد يعيد أكثر من جدول، وهي تستقبل ثلاثة معاملات:

- رقم السجل الذي تريد القراءة بدءا منها.
- أقصى عدد من السجلات تريد قراءته من كل جدول.
- مصفوفة معاملات ParamAray تستقبل مصفوفة جداول DataTable Array، ليمكنك أن ترسل إليها الجداول التي تريد ملأها بالسجلات.

وتعيد هذه الوسيلة عدد السجلات التي تمت إضافتها أو تحديثها في مجموعة البيانات.

**ملء المخطط FillSchema:**

تضيف هذه الفئة صيغتين جديدتين إلى هذه الوسيلة، وهما:

١- الصيغة الأولى تستقبل كائن الجدول DataTable المراد ملؤه، وإحدى قيم المرقم SchemaType التي تعرفنا عليها سابقا.

٢- الصيغة الثانية تستقبل مجموعة البيانات، وإحدى قيم المرقم SchemaType، واسم الجدول في مجموعة البيانات.

وتعيد كلتا الصيغتين كائن الجدول DataTable الذي تم ملؤه بالسجلات.

والفئات التالية ترث الفئة DbDataAdapter:

١. OdbcDataAdapter Class
٢. OleDbDataAdapter Class
٣. SqlDataAdapter Class
٤. OracleDataAdapter Class

وسنكتفي هنا بالتعرف على الفئة SqlDataAdapter.

## فئة موصل بيانات سيكويل SqlDataAdapter Class

هذه الفئة ترث الفئة DbDataAdapter، وهي تعمل كموصل بيانات مخصص للتعامل مع قواعد بيانات سيكويل سيرفر.
ولحدث إنشاء هذه الفئة أربع صيغ:

١- الصيغة الأولى بدون معاملات.
٢- والصيغة الثانية تستقبل أمر التحديد SelectCommand الذي سيستخدمه موصل البيانات لملء مجموعة البيانات.
٣- والصيغة الثالثة تستقبل معاملين:
   - نص جملة التحديد SELECT.
   - كائن الاتصال SqlConnection الذي سيستخدم في الاتصال بقاعدة البيانات.

   لاحظ أن موصل البيانات سيقوم بإنشاء كائن أمر SqlCommand وسيضع جملة التحديد SELECT في الخاصية CommandText الخاصة به، كما سيضع كائن الاتصال في الخاصية Connection الخاصة به.. بعد هذا سيوضع كائن الأمر في الخاصة SelectCommand الخاصة بموصل البيانات.. معنى هذا أن هذه الصيغة تختصر عليك الكثير من الخطوات.
٤- الصيغة الرابعة مماثلة للصيغة السابقة، ولكن معاملها الثاني يستقبل نص الاتصال Connection String اللازم للاتصال بقاعدة البيانات، ليتم استخدامه في إنشاء كائن الاتصال SqlCommand.

وبالإضافة إلى ما ترثه من الفئة الأم من خصائص ووسائل وأحداث، تمتلك هذه الفئة الحدثين التاليين:

### يتم تحديث السجل RowUpdating:

عند استدعاء الوسيلة Update الخاصة بموصل البيانات، فإنها تقوم بالمرور عبر كل سجل في مجموعة البيانات لاستخدامه في تحديث قاعدة البيانات.. وينطلق هذا الحدث قبل استخدام كل سجل موجود في مجموعة البيانات في عملية التحديث.
والمعامل الثاني e لهذا الحدث من النوع SqlRowUpdatingEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | Command | تقرأ أو تغير كائن الأمر SqlCommand الذي سيستخدم في عملية التحديث. |
| | Errors | تعيد كائن الاستثناء Exception الذي يحتوي على معلومات الخطأ الذي حدث في السجل الحالي.. (مفيدة فقط في الحدث RowUpdated). |
| | Row | تعيد كائن صف البيانات DataRow، الذي يتم التعامل معه حاليا في مجموعة البيانات. |
| | Statement Type | تعيد نوع جملة الاستعلام التي سيتم تنفيذها، وهي تعيد إحدى قيم المرقم StatementType التالية:<br>- Select: جملة تحديد.<br>- Insert: جملة إدراج.<br>- Update: جملة تحديث.<br>- Delete: جملة حذف.<br>- Batch: استعلام يتكون من مجموعة أوامر. |
| | Status | تقرأ أو تغير حالة كائن الأمر، وهي تأخذ إحدى قيم المرقم UpdateStatus التالية:<br>- Continue: الاستمرار في تحديث السجل الحالي والسجلات التالية له.<br>- ErrorsOccurred: تطلب من موصل البيانات التعامل مع عملية التحديث كأنها تسببت في حدوث خطأ.. في هذه الحالة سينطلق خطأ في البرنامج فعلا في السطر الذي استدعيت فيه الوسيلة Update الخاصة بموصل البيانات، وعليك معالجة هذا الخطأ بمقطع Try Catch.<br>- SkipCurrentRow: تجاوز السجل الحالي دون استخدامه في تحديث قاعدة البيانات، مع استمرار تحديث باقي السجلات.<br>- SkipAllRemainingRows: تجاوز السجل الحالي والسجلات التالية له وإيقاف عملية التحديث في الحال. |
| | Table Mapping | تعيد كائن خريطة الجدول DataTableMapping، الذي يستخدم للربط بين الجدول في مجموعة البيانات والجدول الأصلي في قاعدة البيانات. |

### تم تحديث السجل RowUpdated:

ينطلق هذا الحدث في كل مرة ينتهي فيها موصل البيانات من استخدام أحد سجلات مجموعة البيانات في عملية التحديث.

والمعامل الثاني e لهذا الحدث من النوع SqlRowUpdatedEventArgs، وهو يمتلك نفس خصائص الفئة SqlRowUpdatingEventArgs التي تعرفنا عليها في الحدث السابق، ويزيد عليها بالعناصر التالية:

| | | |
|---|---|---|
| | RecordsAffected | تعيد عدد السجلات التي تأثرت بعملية التحديث في قاعدة البيانات.. هذا العدد يكون صفرا إذا لم يتم العثور على السجل، أو حدث خطأ، ويكون -١ إذا كنت تستخدم جملة SELECT الخاصة بأمر التحديد.. لاحظ أن جملة SELECT الموجودة في نهاية أمر التحديث وأمر الإدراج تؤثر على قيمة هذه الخاصية، وتجعل قيمتها صفرا! |
| | RowCount | هذه الخاصية مفيدة إذا كانت للخاصية UpdateBatchSize الخاصة بموصل البيانات قيمة أكبر من ١، ففي هذه الحالة يقوم موصل البيانات بتحديث أكثر من سجل في المرة الواحدة، حيث تعيد إليك هذه الخاصية عدد السجلات التي تم التعامل معها.. لاحظ أن هذا العدد قد يكون مساويا لقيمة الخاصية UpdateBatchSize أو أصغر منها (في حالة عدم وجود سجلات كافية في مجموعة البيانات). |
| | CopyToRows | إذا كانت للخاصية UpdateBatchSize قيمة أكبر من ١، فإن الخاصية e.Row لا تفيدك لمعرفة السجلات التي تم حفظ بياناتها في قاعدة البيانات.. وبدلا منها، يمكنك إرسال مصفوفة صفوف DataRow Array إلى هذه الوسيلة، لتوضع فيها صفوف مجموعة البيانات |

| | | |
|---|---|---|
| | | التي تم استخدامها في عملية التحديث، مع ملاحظة أن هذه العملية مرجعية By Reference، أي أن أي تغيير في الصفوف الموجودة في المصفوفة، سيظهر تأثيره في مجموعة البيانات.. ويجب أن تحتوي المصفوفة المرسلة على الأقل على عدد من الخانات يساوي قيمة الخاصية e.RowCount.<br>وتوجد صيغة أخرى لهذه الوسيلة، لها معامل ثان، يستقبل رقم الخانة، الذي سيتم وضع الصفوف في المصفوفة بدءا منه. |

| ملحوظة: |
|---|
| لعلك تتساءل لماذا لا يمتلك معامل الحدث RowUpdating الوسيلة CopyToRows.. السبب في هذا أن الحدث RowUpdating ينطلق دائما قبل تحديث كل صف على حدة، حتى لو كان موصل البيانات سيستخدم مجموعة أوامر SQL Batch لتحديث مجموعة صفوف دفعة واحدة.. هذا منطقي، لأن موصل البيانات يقرأ سجلا تلو سجل من مجموعة البيانات (ويطلق الحدث RowUpdating لكل سجل)، وبعد هذا يكوّن موصل البيانات مجموعة أوامر لتحديث السجلات التي قرأها، ويرسل هذه الأوامر المجمعة إلى قاعدة البيانات، ثم يطلق الحدث RowUpdated بعد تنفيذها. |

لاحظ أنك لو لم تكتب الإجراء المستجيب للحدث RowUpdated، فإن الوسيلة Update تطلق خطأ في البرنامج إذا حدثت مشكلة تطابق Concurrency Violation عند حفظ أحد السجلات.. أما إذا كتبت الإجراء المستجيب لهذا الحدث، فإن الخطأ لا يحدث، وتتاح لك الفرصة لكتابة الكود الذي يحل مشكلة التطابق.

لكن.. ما هو موضوع التطابق Concurrency هذا؟

هذا هو موضوع الفقرة التالية.

## التصارع على تحديث البيانات:

هناك مشكلة رئيسية ستواجهك عند التعامل مع قاعدة بيانات يستخدمها أكثر من موظف في نفس الوقت، وهي التضارب بين التعديرت التي يجريها أكثر من موظف على نفس السجل.. تخيل هذه الحالة:

- قام مستخدم برنامجك بتحميل سجلات الكتب، وقام بتعديل سعر كتاب "عضا الحكيم" من ٥ جنيهات إلى ٧ جنيهات.
- عند محاولة برنامجك حفظ هذه التغييرات في قاعدة البيانات، كان مستخدم آخر قد غير عدد النسخ المتاحة المتاحة من كتاب "عصا الحكيم" من ٢٠٠٠ إلى ٣٠٠٠.

السؤال الآن هو: ماذا نفعل في هذه الحالة؟

لو حفظ برنامجك سجل الكتاب "عصا الحكيم" فسيعدل سعره إلى ٧ جنيهات، لكنه سيعيد عدد النسخ المتاحة منه إلى ٢٠٠٠!

أما لو أبقينا على التعديلات التي أجراها المستخدم الآخر، فهذا معناه الإبقاء على التعديل الذي حدث في عدد النسخ، لكن السعر سيظل ٥ جنيهات!

طبعا في كلتا الحالتين ستحدث مشكلة في العمل.. وعندما سيحاول المدير معاقبة مسئول المخازن في الحالة الأولى فسيقسم له بأغلظ الأيمان إنه غير عدد النسخ المتاحة، وعندما سيحاول معاقبة مسئول المبيعات في الحالة الثانية، فسيقسم له إنه غير ثمن النسخة، وكلاهما صادق في قسمه، وأنت الذي خربت بيته!

تسمّى هذه المشكلة باسم مشكلة التطابق Concurrency Problem.. ويمكن علاجها بأحد الحلين التاليين:

### ١- التطابق المتشائم Pessimistic Concurrency:

استخدم هذا الحل لو كنت "متشائما" بخصوص تعارض البيانات التي يحفظها المستخدمين، أو كان أي تعارض يمكن أن يؤدي إلى خسائر كبيرة للعمل الذي ينظمه برنامجك، حيث إن التطابق المتشائم يمنع حدوث أي تضارب في البيانات، وذلك بإغلاق Lock سجلات قاعدة البيانات قام أي مستخدم بتحميلها، مما يمنع أي مستخدم آخر من تغييرها إلى أن يتم يغلق المستخدم الأول الاتصال وتتم إزالة الإغلاق.. ويمكن تنفيذ هذا الحل في دوت نت باستخدام التعاملات Transactions، لهذا سنؤجل تطبيقه إلى الكتاب القادم إن شاء الله.

وعند استخدام هذا الحل، تكون جملة التحديث بسيطة للغاية، لأنك تستخدم المفتاح الأساسي للحقل للعثور عليه في قاعدة البيانات، ومن ثم تغير قيمه مباشرة، لأنك واثق أنه لم يتغير منذ أن قمت بتحميله.. هكذا مثلا ستكون جملة تحديث سجلات المؤلفين:

```
UPDATE Authors
SET Author = @Author,
      CountryID = @CountryID,
      Phone = @Phone,
      About = @About
WHERE ID = @ID;
```

لاحظ أن المعاملات الموجودة في هذا الاستعلام تأخذ قيمها من خانات السجل الذي يتم تحديثه في مجموعة البيانات.

ويعتبر التطابق المتشائم حلا حاسما للمشكلة، لأن أي مستخدم آخر سيحاول تعديل السجلات المتنازع عليها سيحصل على رسالة خطأ تخبره بأنها مغلقة من قبل مستخدم آخر.. وفي هذه الحالة عليك أن تجعل برنامجك ينتظر انتهاء الإغلاق، ومن ثم يعرض للمستخدم السجلات التي يحاول تحديثها، ليتعرف على التغييرات التي تمت عليها، ومن ثم يقرر كيف يوائم بينها وبين التغييرات التي أجراها، ثم يعيد حفظها في قاعدة البيانات.

لكن المشكلة هي أن التطابق المتشائم سيهبط بكفاءة البرنامج إذا استمر إغلاق كل سجل لفترات زمنية طويلة، أو إذا تم تحديث أعداد ضخمة من السجلات على التتابع، وذلك لأن إغلاق السجلات يستهلك جزءا من وقت تشغيل وذاكرة الخادم، كما أنه يحتاج إلى إبقاء قنوات الاتصال مفتوحة مع المستخدمين الذين قاموا بعملية الإغلاق، مما يحرم مستخدمين آخرين من الاتصال بقاعدة البيانات في ذلك الوقت.. لكن يظل التطابق المتشائم الحل الأفضل عند التعامل مع قاعدة بيانات يتصل بها عدد كبير من المستخدمين في نفس اللحظة، ويتصارعون على تحديث نفس السجلات، لأن استخدام كثرة عمليات التراجع عن التعاملات Transactions Rollback لاستعادة القيم الأصلية قبل التضارب، تستهلك الخادم في هذه الحالة بأكثر مما تفعل عمليات الإغلاق.

## ٢- التطابق المتفائل Optimistic Concurrency:

هذا هو الحل المفضل والأسهل في تقنية ADO.NET، وقد سمي بهذا الاسم لأنه يفترض أن المستخدمين لن يحاولوا تعديل قاعدة البيانات، أثناء تعاملك مع بياناتها في برنامجك، إلا في حالات نادرة.

لكن ماذا يحدث لو حدث التعارض فعلا؟.. كيف نحل المشكلة في هذه الحالة؟ في الحقيقة هناك عدة حلول متبعة، وعليك اختيار لحل الذي يناسبك منها تبعا لما يناسبك.. وهذه الحلول هي:

أ. عند تحديث السجل، يتم البحث عنه في قاعدة البيانات بواسطة قيم كل حقوله، وليس فقط بواسطة المفتاح الأساسي.. هذه مثلا جملة الاستعلام الخاصة بتحديث سجلات المؤلفين:

```
UPDATE Authors
SET Author = @Author,
    CountryID = @CountryID,
    Phone = @Phone,
    About = @About
WHERE (ID = @Original_ID) AND
    (Author = @Original_Author) AND
    (CountryID = @Original_CountryID) AND
    (@IsNull_Phone = 1) AND (Phone IS NULL) OR
    (ID = @Original_ID) AND
    (Author = @Original_Author) AND
    (CountryID = @Original_CountryID) AND
    (Phone = @Original_Phone)
```

هذا الاستعلام يضمن لك أنه لو حدث أي تغيير في السجل من قبل مستخدمين آخرين، فلن يعثر عليه برنامجك، وبالتالي لن يتم حفظ التغييرات التي قام بها مستخدم برنامجك.

لعلك تلاحظ في هذا الاستعلام وجود معاملين للتعامل مع كل حقل.. مثلا، يتم التعامل مع حقل المؤلفين من خلال المعاملين @Author و @Original_Author.. فما هو الفارق بينهما؟

لكي تفهم هذا الفارق، عليك أن تعرف أن مجموعة البيانات DataSet تحتفظ بنسختين من كل سجل يتم وضعه فيها:

- النسخة الأصلية Original Version:
وهي تحتوي على بيانات السجل الأصلي كما تم تحميلها من قاعدة البيانات، لاستخدامها بعد ذلك في البحث عن السجل الأصلي في قاعدة البيانات لتحديثه.
- النسخة الحالية Current Version:
وهي تحتوي على بيانات السجل بعد التغييرات التي أجراها المستخدم عليها، وذلك لاستخدامها في تحديث السجل الأصلي في قاعدة البيانات.

وكل ما يفعله استعلام التحديث السابق، هو تعريف معاملين لكل حقل، أحدهما يقرأ قيمته الحالية (مثل @Author) ويتم استخدامه لحفظ التغييرات في قاعدة البيانات، والآخر يقرأ قيمته الأصلية

(مثل @Original_Author) ويستخدم للبحث عن السجل الأصلي في قاعدة البيانات.. ويتم التفريق بين هذين المعاملين باستخدام الخاصية SourceVersion الخاصة بكائن المعامل DataParameter، والذي يمكن إرسال قيمته من خلال المعامل التاسع لحدث الإنشاء New.. هكذا مثلا يتم تعريف المعامل @Author.. لاحظ أننا لن نرسل قيمة المعامل SourceVersion، لهذا سيتم سيقرأ القيمة الحالية افتراضيا:

```
Dim P1 As New SqlParameter("@Author",
    SqlDbType.NVarChar, 0, "Author")
```

وهكذا يتم تعريف المعامل @Original_Author:

```
Dim P2 As New SqlParameter("@Original_Author",
    SqlDbType.NVarChar, 0,
    ParameterDirection.Input, False, 0, 0, "Author",
    DataRowVersion.Original, Nothing)
```

لكن.. لماذا لا يوجد شرط على الحقل About في المقطع Where؟

السبب في هذا أننا عرفنا هذا الحقل من النوع nvarchar(MAX)، وهذا معناه أنه يتسع لنص قد يصل إلى ٢ مليار حرف، وهذا حجم هائل، وستكون مقارنة هذا الحقل مضيعة للوقت.. لكن لو كنت مصرا، فيمكنك تعديل الاستعلام.. لا أنصحك بفعل هذا من نافذة الخصائص، لأنها ستعجز عن إنشاء المعامل @Original_About بشكل صحيح، وبلا من هذا يمكنك إضافة هذا الكود في بداية حدث تحميل النموذج:

```
' إضافة شرط إلى نهاية استعلام التحديث
' سيحدث خطأ لو كان هناك استعلام تحديد في نهاية استعلام التحديث
DaAuthors.UpdateCommand.CommandText &=
    " And About = @Original_About"
' تعريف معامل جديد
Dim P As New SqlParameter("@Original_About",
    SqlDbType.NVarChar, -1,
    ParameterDirection.Input, False, 0, 0, "About",
    DataRowVersion.Original, Nothing)
' إضافة المعامل إلى مجموعة معاملات أمر التحديث
DaAuthors.UpdateCommand.Parameters.Add(P)
```

أو يمكنك فتح ملف تصميم النموذج Form1_Designer.vb، وتعديل استعلام التحديث مباشرة، وإن كنت لا أنصح بهذا.
لاحظ أن من الأفضل تغيير نوع الحقل About ليكون أكثر ملاءمة لوظيفته.. يمكنك افتراض أن أطول نبذة لا تزيد عن ٥٠٠ حرف مثلا، وتعريف هذا الحقل منن النوع nvarchar(50).
دعنا نعد إلى استعلام التحديث السابق، فمازال هناك نوع ثالث من المعاملات لم نتطرق إليه.. هذا المعامل مخصص للتعامل مع القيم المنعدمة NULL (مثل المعامل @IsNull_Phone).. وسبب احتياجنا إلى هذا المعامل، هو أن أي عملية مقارنة مع خانة منعدمة تعطي دائما False، لهذا لو كانت خانة الهاتف فارغة في قاعدة البيانات، وكانت فارغة أيضا في النسخة الأصلية من السجل، فإن مقارنتهما ستعطي False، وهذا يعني أن برنامجك لن يستطيع تحديث خانة الهاتف أبدا!
لحل هذه المشكلة، نستخدم الشرط التالي:

```
(@IsNull_Phone = 1) AND (Phone IS NULL)
OR (ID = @Original_ID)
```

هذا الشرط يتأكد من أن خانة الهاتف فارغة في مجموعة البيانات، وأنها فارغة أيضا في قاعدة البيانات، أو أن الخانتين فيهما قيمتين متساويتين.
ويتم تعريف المعامل @IsNull_Phone بوضع القيمة True في الخاصية SourceColumnNullMapping الخاصة بكائن المعامل، وهو ما يمكن فعله بإرسال القيمة True إلى المعامل التاسع في إحدى صيغ حدث الإنشاء New كالتالي:

```
Dim P3 As New SqlParameter("@IsNull_Phone",
      SqlDbType.Int, 0, ParameterDirection.Input,
      0, 0, "Phone", DataRowVersion.Original,
      True, Nothing, "", "", "")
```

لاحظ أن موصل البيانات يستخدم استعلام التحديث السابق بصورة افتراضية، لكن هذا قد يهبط بكفاءة برنامجك، إذا كان الجدول يحتوي على عدد كبير من السجلات، مما يعقد استعلام التحديث، ويستهلك وقتا ملموسا من سيكويل سيرفر للبحث عن السجل في قاعدة البيانات، لأنه سيقارن هنا كل الخانات، وليس من المتوقع وجود فهارس لكل أعمدة قاعدة البيانات.

ب. عند تحديث السجل، يتم البحث عنه في قاعدة البيانات بواسطة مفتاحه الأساسي فقط (كما فعلنا في التطابق المتشائم).. ميزة هذه الطريقة أنها تبسط استعلام التحديث، وتجعل العثور على السجل في قاعدة البيانات أسرع لأن المفتاح الأساسي مفهرس Indexed، وهي ميزة هائلة في قواعد البيانات الضخمة.. لكن عيب هذه الطريقة هي أنها تستخدم مبدأ "آخر تحديث يكسب!".. حيث إنّ السجلات يتمّ حفظها إلى قاعدة البيانات، حتّى ولو كانت هناك تعديلات قد أجراها مستخدم آخر عليها.. إنك تفرض سجلاتك على قاعدة البيانات رغم أنف الجميع (وهذا سيخرب بيت مدير المخازن عند تعديل سعر كتاب عصا الحكيم).. لكن أحيانا تكون هذه الطريقة مقبولة، كما في أنظمة حجز رحلات الطيران، لأنّ آخر تعديل في مواعيد الحجز هو الأولى بالاعتبار.

ج. عند تحديث السجل، يتم البحث عنه في قاعدة البيانات بواسطة المفتاح الأساسي وطابع الوقت Timestamp.. لفعل هذا عليك إضافة عمود من النوع timestamp إلى الجدول.. هذا العمود يتغير تلقائيا كلما تم تعديل السجل، وبهذا لو كان طابع الوقت الذي تحتفظ به مجموعة البيانات مختلفا عن طابع الوقت الموجود في قاعدة البيانات، فهذا معناه أن السجل قد تغير، وفي هذه الحالة لن يحفظ برنامجك التغييرات في هذا السجل.. لاحظ أن المعالج السحري لموصل البيانات ينتج أوامر تحديث تعتمد على طابع الوقت إذا وجده في استعلام التحديد، أما إذا لم يجده، فإنه ينتج أوامر تحديث تقارن كل الحقول كما في الطريقة أ.

لاحظ أن الطريقتين أ و ج هما الأكثر شيوعا، لكن بهما مشكلة كبيرة، وهي أن برنامجك سيكتفي بعرض رسالة خطأ للمستخدم تخبره بأن أحد الصفوف يتعارض مع قاعدة البيانات، دون أن يعرف التغييرات التي حدثت في قاعدة البيانات، ودون أن يستطيع إعادة حفظ السجل، لأن نفس الخطأ سيستمر في الحدوث!!

ولحل هذه المشكلة، عليك استخدام الحدث RowUpdated الخاص بموصل البيانات بالطريقة التالية:

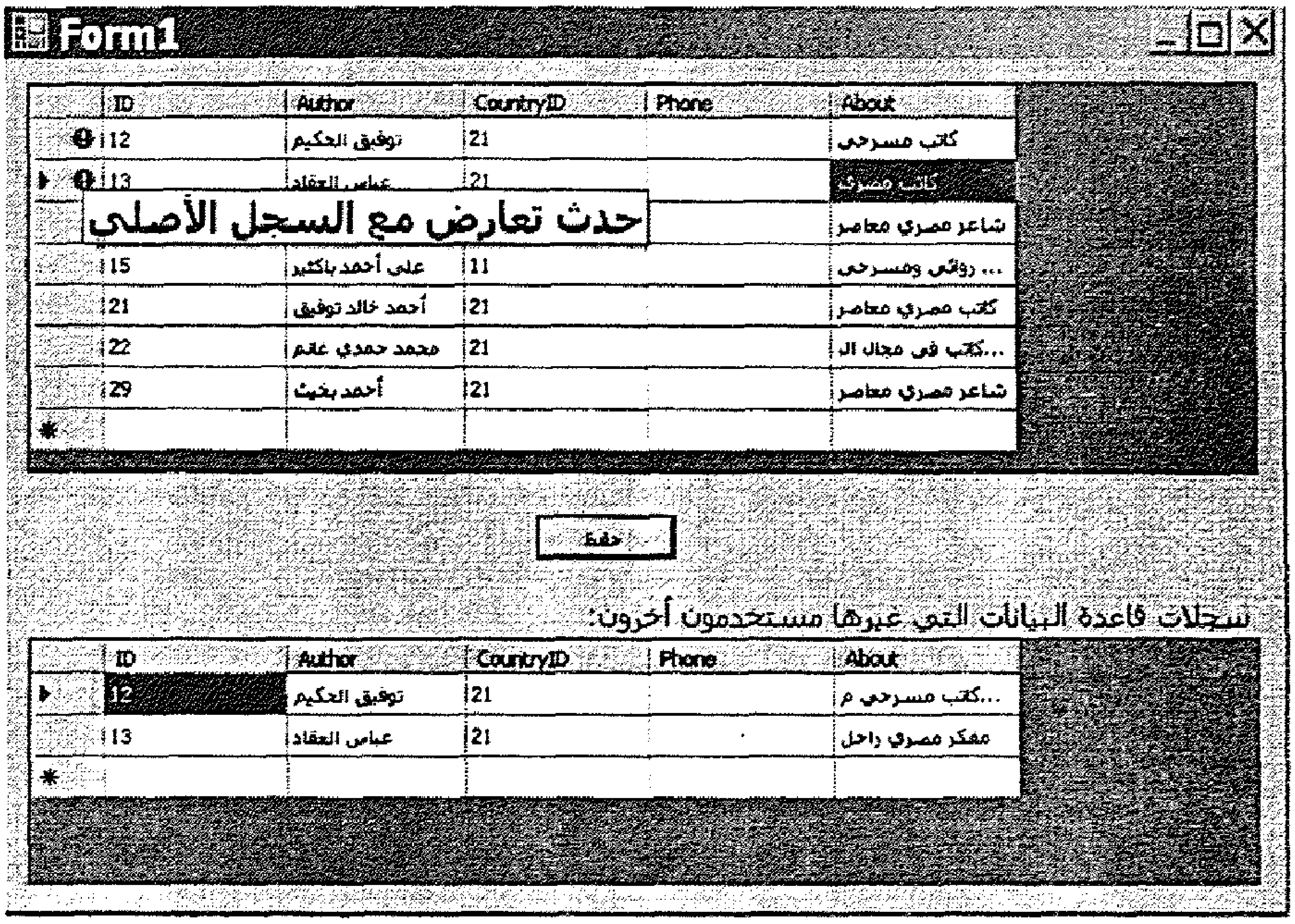
- إذا كانت قيمة الخاصية e.RecordsAffected تساوي صفرا، فهذا معناه أن أمر التحديث لم يؤثر على قاعدة البيانات، لأنه لم يجد السجل المطلوب تحديثه، إما لأن مستخدما آخر حذفه أو عدل بياناته.. هذا هو التعارض الذي نبحث عنه.. لاحظ أن وجود جملة SELECT في نهاية أمر التحديث سيجعل الخاصية RecordsAffected تعيد الرقم ٠ دائما.. لهذا إذا أردت أن تستفيد

من هذه الخاصية في معرفة إن كان التحديث قد تم أم لا، فعليك أن تزيل جملة التحديد من نهاية أمر التحديث.. وسنعرف كيف نفعل هذا من خلال المعالج السحري لموصل البيانات بعد قليل.

- ضع نصا يدل على حدوث خطأ في الخاصية e.Row.RowError، مثل "حدث تعارض مع السجل الأصلي لأن أحد المستخدمين قام بتعديله أو حذفه".. هذا سيجعل أيقونة الخطأ تظهر بجوار السجل في جدول العرض، وعند التحليق فوقها بالفارة سيظهر تلميح على الشاشة يعرض للمستخدم النص الذي كتبته في هذه الخاصية.
- إذا أردت أن يحدث خطأ في البرنامج في سطر استدعاء الوسيلة Update لتعالجه بطريقتك الخاصة، فضع في الخاصية e.Status القيمة ErrorsOccurred.. أما إذا أردت مواصلة عملية التحديث، فضع فيها القيمة Continue أو SkipCurrentRow.. دعنا نستخدم القيمة الأخيرة.
- يبدو الأمر رائعا حتى الآن، وسيلاحظ المستخدم ظهور أيقونات الخطأ بجوار السجلات التي فشل تحديثها.. لكن المشكلة أن المستخدم لا يعرف التعديل الذي أدخله المستخدمون الآخرون على السجل الأصلي في قاعدة البيانات.. لهذا سنلجأ إلى طريقة مبتكرة، وهي استخدام موصل بيانات اسمه DaErrAuthor لتحميل السجل الأصلي مرة أخرى في مجموعة بيانات خاصة اسمها DsErr، وعرضه في جدول عرض آخر اسمه DgErrors، ليقارن المستخدم بين البيانات التي يريد حفظها، والبيانات التي حفظها مستخدم آخر، ويتخذ قراره بناء على هذا، كما هو موضح في الصورة.

لاحظ أن السجلات التي حذفها مستخدم آخر من قاعدة البيانات لن تظهر في جدول السجلات المعدلة.. من السهل أن يفهم المستخدم أن السجل قد حذف إذا لم يجده، لكننا أيضا نستطيع التسهيل عليه، بتغيير رسالة الخطأ إذا كان استعلام التحديث لا يعيد أية سجلات، وذلك كالتالي:

```
If DaErrAuthor.Fill(DsErr, "Authors") > 0 Then
   e.Row.RowError = "حدث تعارض مع السجل الأصلي" &
          " لأن أحد المستخدمين قام بتعديله أو حذفه "
Else
   e.Row.RowError = "هذا السجل حذفه مستخدم" &
          " آخر من قاعدة البيانات "
End If
```




- بقيت أمامنا خطوة أخيرة، وهي: كيف نسمح للمستخدم بتعديل السجل المعدل أو إعادة السجل المحذوف، إن قرر هذا؟.. أنا أرى أن أفضل حل، هو عرض قائمة موضعية حينما يضغط الصف الذي به خطأ في الجدول العلوي، ومن هذه القائمة يختار ما يناسبه مما يلي:

أ. الأمر "أريد حفظ تعديلاتي":

سنستخدم هذا الأمر عندما يغير مستخدم آخر السجل، وهو ينسخ القيم الأصلية من سجل قاعدة البيانات المعدل ويجعلها القيم الأصلية للسجل الخاص بالمستخدم، وهذا حتى ينجح استعلام التحديث في العثور على السجل الأصلي في قاعدة البيانات، ومن ثم لو ضغط المستخدم زر الحفظ يتم حفظ تعديلاته.. والمستخدم هو المسئول عن نقل أية قيمة يدويا من السجل الأصلي المعروض في جدول العرض السفلي إلى السجل الخاص به قبل ضغط زر الحفظ.. والسجل الذي تنجح محاولة حفظه مرة أخرى، علينا أن نزيل سجل الخطأ المناظر له من جدول العرض السفلي.

لاحظ أنك في البرامج العملية قد تحتاج إلى مراعاة أولويات المستخدمين في إجراء التغيير.. مثلا: لو كان المدير هو من قام بتعديل السجل، فسيستشيط غضبا لو قام أحد الموظفين بإلغاء تعديله!.. لهذا قد تحتاج إلى إضافة حقل اسمه UserID إلى الجدول، ليربطه بجدول المستخدمين Users، بحيث تضع رقم

المستخدم الذي أجرى آخر تعديل، وعند حدوث التعارض في البرنامج، لا تسمح للمستخدم باتخاذ قرار حفظ تعديلاته إلا إذا كان المستخدم الآخر أقل أولوية منه أو على الأقل له نفس الأولوية في إجراء التعديلات.. ويمكنك معرفة أولويات المستخدمين من الجدول User، الذي لا بد أن يحتوي على عمود يوضح وظيفة المستخدم، أو عمود يوضح ترتيبه في السلم الوظيفي أو مدى صلاحياته.

ب. إلغاء تعديلاتي:
سنستخدم هذا الأمر عندما يغير مستخدم آخر السجل.. هذا الأمر مفيد عندما يقرر مستخدم البرنامج إلغاء تعديلاته هو، وكل ما سنفعله هو نقل السجل المعدل إلى مجموعة البيانات ليحل محل السجل الذي عدله مستخدم البرنامج، مما يلغي تعديلاته، ويحافظ على التعديل القادم من قاعدة البيانات.

ج. الأمر "أريد إعادة إدراج السجل المحذوف":
سنستخدم هذا الأمر إذا حذف مستخدم آخر السجل من قاعدة البيانات.. وكل ما يفعله هذا الأمر، هو تغيير حالة السجل الحالي إلى Added، ليعتبره أمر التحديث سجلا جديدا ويضيفه إلى قاعدة البيانات.

د. الأمر "إزالة السجل المحذوف":
سنستخدم هذا الأمر إذا حذف مستخدم آخر السجل من قاعدة البيانات.. وكل ما يفعله هذا الأمر، هو حذف السجل الخاص بالمستخدم من مجموعة البيانات.

وستجد الكود الكامل الذي ينفذ كل هذه الأفكار في المشروع OptimisticConcurrency.. وتوجد نسخة أخرى منه في المشروع OptimisticConcurrencyWithTimeStamp، نستخدم فيها طابع الوقت، حيث عرفنا عمودا اسمه RowVersion في جدول المؤلفين نوعه Timstamp.. لاحظ أن عرض طابع الوقت في جدول العرض DatagridView يسبب أخطاء لأنه يحاول رسم طابع الوقت باعتباره صورة!.. ولحل المشكلة، عليك إخفاء عمود طابع الوقت، فلا يوجد مبرر أصلا لعرضه للمستخدم!.. لفعل هذا استخدمنا الجملة التالية:

```
DgAuthors.Columns("RowVersion").Visible = False
```

لاحظ أن هناك مشكلة ستواجهنا في هذا البرنامج، بسبب عدم استخدامنا جملة تحديد Select بعد جملة التحديث Update، وذلك لأن طابع الوقت يتغير في قاعدة البيانات باستمرار بعد كل عملية تحديث، ولو لم ننعش مجموعة البيانات بالقيم الجديدة له، فستحدث مشكلة تطابق بلا داع.. ولحل هذه المشكلة، استخدمنا موصل بيانات اسمه DaTimestamp، مهمته الحصول على السجل الذي تم تحديثه، ووضعه في مجموعة البيانات لإنعاشها.. لهذا يستخدم أمر التحديد الخاص بهذا الموصل جملة التحديد التالية:

```
Select * From Authors
Where ID = @ID
```

وأنسب مكان لاستخدام هذا الموصل، هو الحدث RowUpdated، لأنه ينطلق مباشرة بعد تحديث الصف، لهذا يمكننا أن نقرأ الصف مرة أخرى بعد أن غيرت قاعدة البيانات طابع الوقت الخاص به.. لهذا طورنا جملة الشرط التي نستخدمها في هذا الحدث، بإضافة المقطع Else كالتالي:

```
If e.RecordsAffected = 0 Then
  ' الكود المناسب لحل مشكلة التطابق
ElseIf e.StatementType <> StatementType.Delete Then
  TimestampCmd.Parameters(0).Value = e.Row("ID")
  DaTimestamp.Fill(Ds, "Authors")
End If
```

لاحظ أننا استخدمنا شرطا لاستثناء حالة حذف سجل، فالسجل سيحذف من مجموعة البيانات كما حذف من قاعدة البيانات، وليست لدينا مشكلة.

غير هذا لن تجد أي اختلاف في كود هذا المشروع عن المشروع OptimisticConcurrency، فالفروق كلها تنحصر في صيغة استعلامات التحديث، التي تزيد كفاءة برنامجك بسبب استخدامها لطابع الوقت، بديلا عن مقارنة كل القيم الموجودة في خانات الصف.

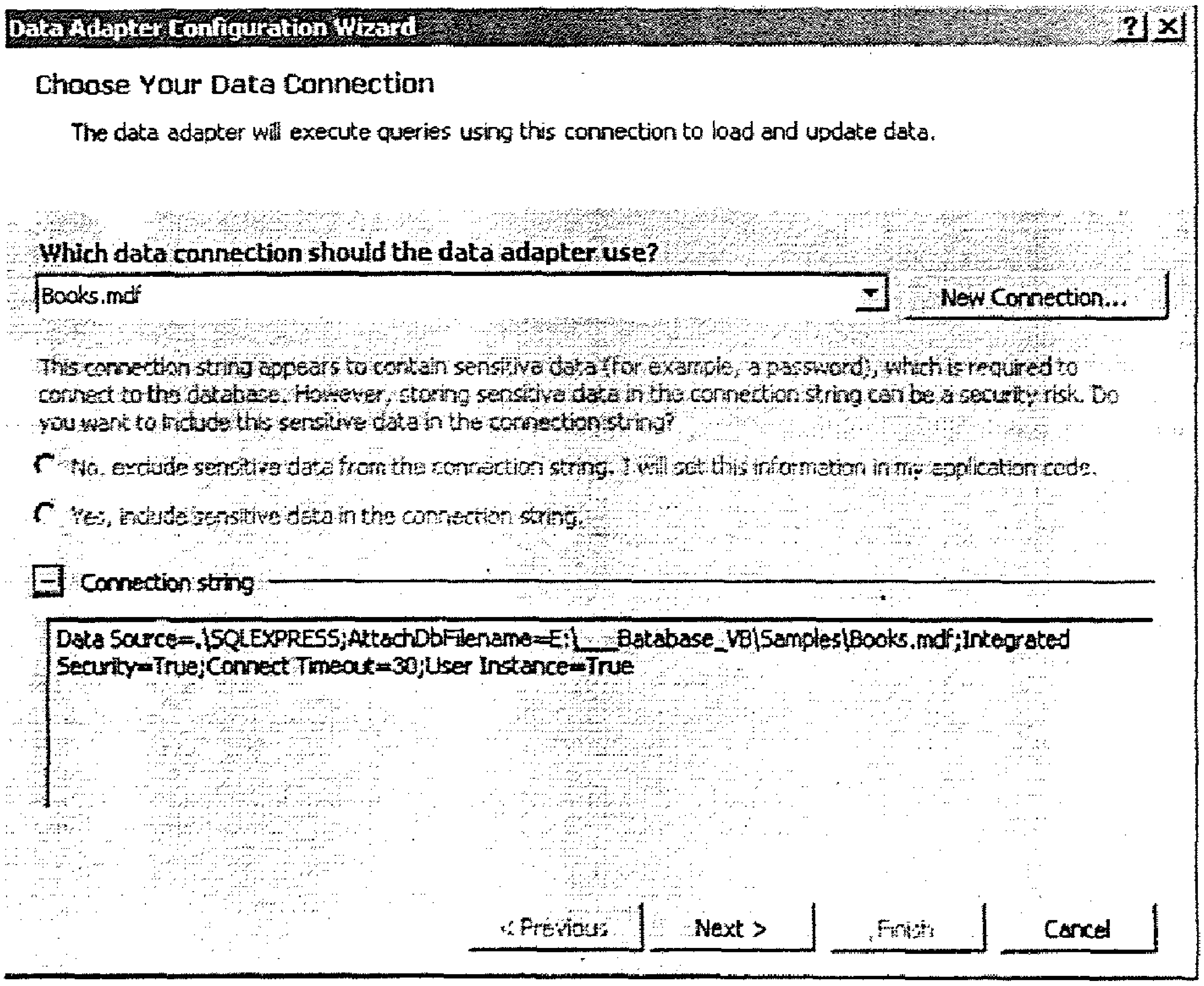
## المعالج السحري لإعداد موصّل البيانات

## Data Adapter Configuration Wizard

استخدم هذا المعالج لتسهيل ضبط وظيفة وخصائص موصل البيانات.. ويمكنك تشغيل هذا المعالج باتباع أي مما يلي:

- النقر مرتين على أيقونة موصل البيانات SqlDataAdapter في صندوق الأدوات ToolBox.
- سحب أيقونة موصل البيانات من صندوق الأدوات وإلقائها على النموذج.
- ضغط موصّل البيانات بعد إضافته إلى صينية مكونات النموذج Component Tray بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة ضغط الأمر Configure Data Adapter.

ويبدأ هذا المعالج بنافذة تطلب منك اختيار قاعدة البيانات التي تريد الاتصال بها:



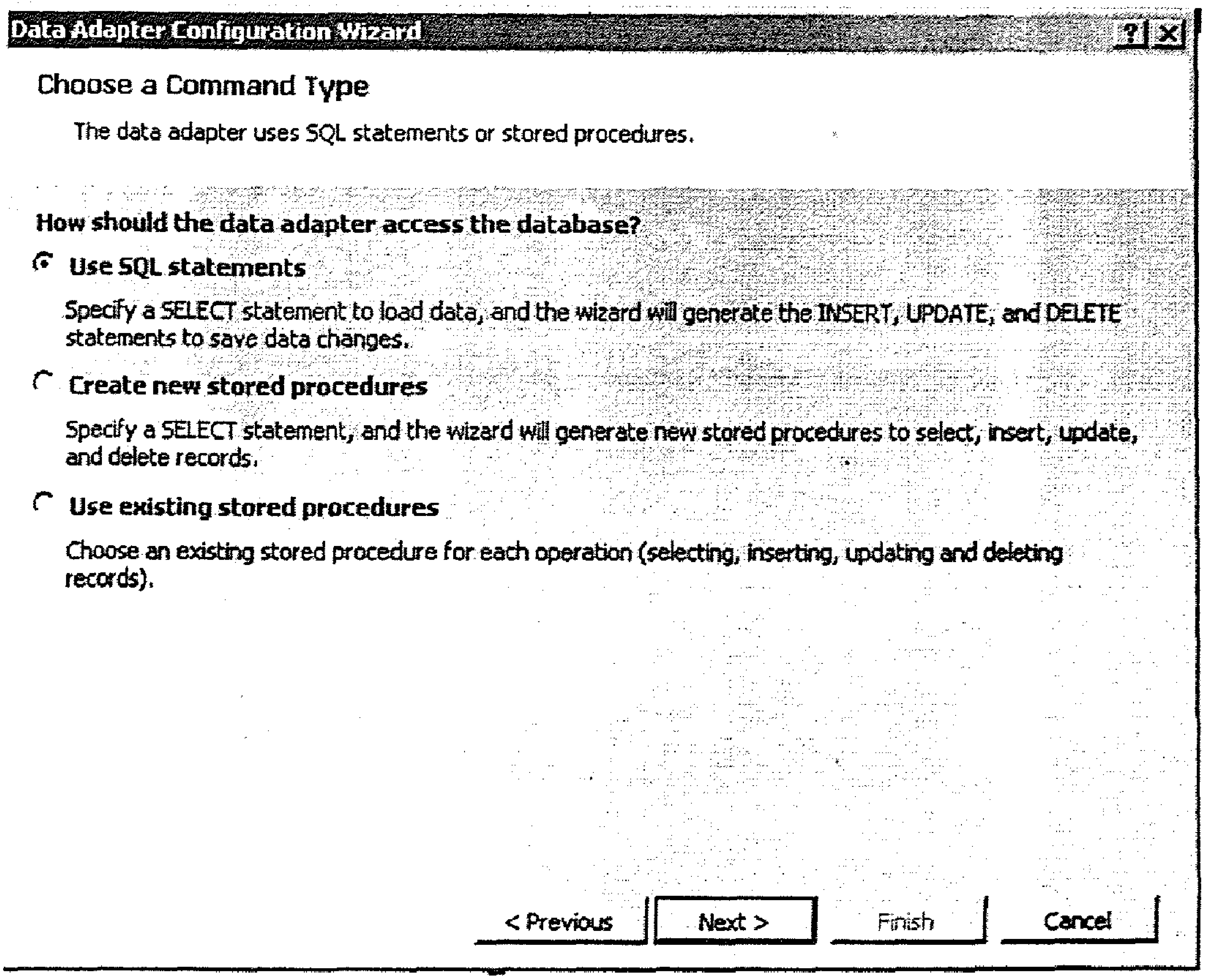

في هذه النافذة، يمكنك استخدام القائمة المنسدلة لاختيار اسم إحدى قواعد بيانات سيكويل سيرفر التي أضفت اتصالا بها من قبل في متصفح الخوادم Server Explorer.

ولو أردت إنشاء اتصال جديد بقاعدة بيانات أخرى، فاضغط الزرّ New Connection لتظهر لك نافذة إضافة اتصال Add Connection التي تعرفنا عليها من قبل في متصفح الخوادم.

ولو ضغطت العلامة + المجاورة للجملة Connection String في الجزء السفلي من النافذة، فسيتم عرض مربع نص قابل للقراءة فقط، به نص الاتصال بقاعدة البيانات التي اخترتها.. ويمكنك نسخ هذا النص لاستخدامه في أي موضع آخر في البرنامج لو أردت.

جرب على سبيل المثال اختيار قاعدة بيانات الكتب Books.mdf، واضغط الزر Next.

في النافذة التالية يمكنك اختيار نوع الاستعلام الذي ستستخدمه لإحضار البيانات من قاعدة البيانات:



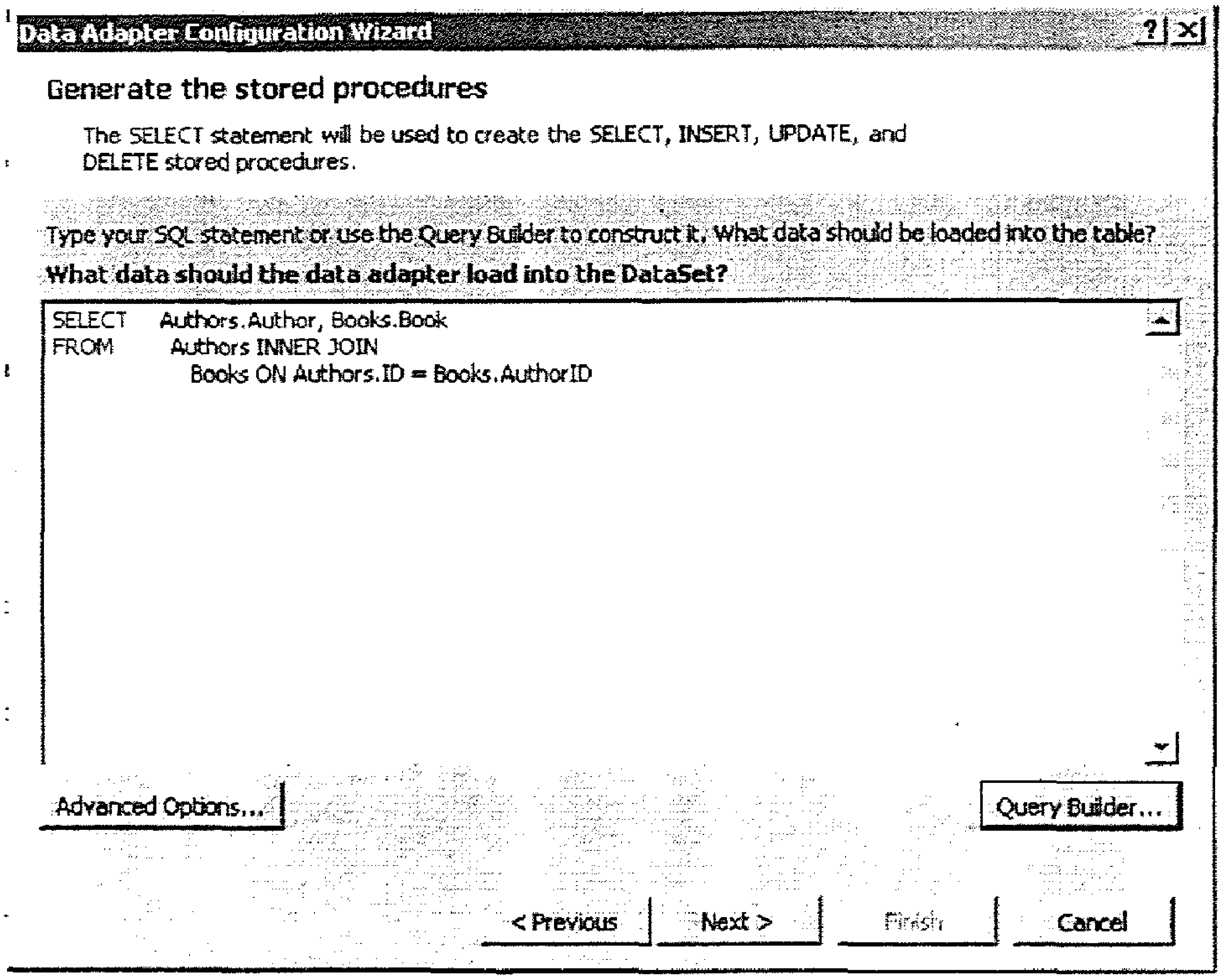

هذه النافذة تتيح لك اختيار واحد مما يلي:

- استخدام جمل SQL (Use SQL Statements).
- إنشاء إجراءات مخزّنة جديدة (Create New Stored Procedures.).

o استخدام إجراءات مخزّنة موجود سابقا في قاعدة البيانات (Use Existing Stored Procedures).

ويحدد اختيارك، النافذة التالية التي ستظهر عندما تضغط الزر Next، وذلك كالتالي:

- إذا اخترت استخدام جملة SQL أو إنشاء إجراء مخزن جديد، فستظهر لك النافذة التالية:




هذه النافذة تقدم لك مربع نص تستطيع أن تضيف فيه يدويا أو باللصق، نص جملة الاستعلام أو جملة إنشاء الإجراء المخزن الجديد.

ويمكنك ضغط الزرّ Query Builder لاستخدام باني الاستعلام في إنشاء جملة SQL، وعند إغلاق باني الاستعلام ستجد هذه الجملة مضافة إلى مربع النص.. لاحظ أن هذا الزر موجود أيضا في حالة إنشاء إجراء مخزن جديد، لأنك قد تحتاج إلى إضافة جملة استعلام داخل الإجراء المخزن، لهذا من الأسهل أن تنشئ هذه الجملة بباني الاستعلام، ثم تضيف إلى مربع النص بعد هذا صيغة الإجراء المخزن الذي يحتويها.. وإن كنت أنصحك بعدم إنشاء الإجراءات المخزنة بهذه الطريقة، لأن إنشاء الإجراء المخزن باستخدام متصفح الخوادم Sever Explorer أسهل وأفضل تنسيقا، ويتيح لك ايضا استخدام باني الاستعلام، مع مراجعة صياغة جمل الإجراء المخزن، واختبار نتائجه.

**ملحوظة:**

لا يسمح المعالج السحري بكتابة أكثر من جملة SQL مفصولة بالعلامة ; جرب مثلا استخدام الجملة التالية لملء مجموعة البيانات بجدولي المؤلفين والكتب كاملين:

**SELECT * FROM Authors;**
**SELECT * FROM Books**

لو ضغطت الزر Next فستظهر نافذة تخبرك بوجود خطأ في جملة الاستعلام، وسترفض مواصلة الخطوات ما لم تصحح هذا الخطأ.

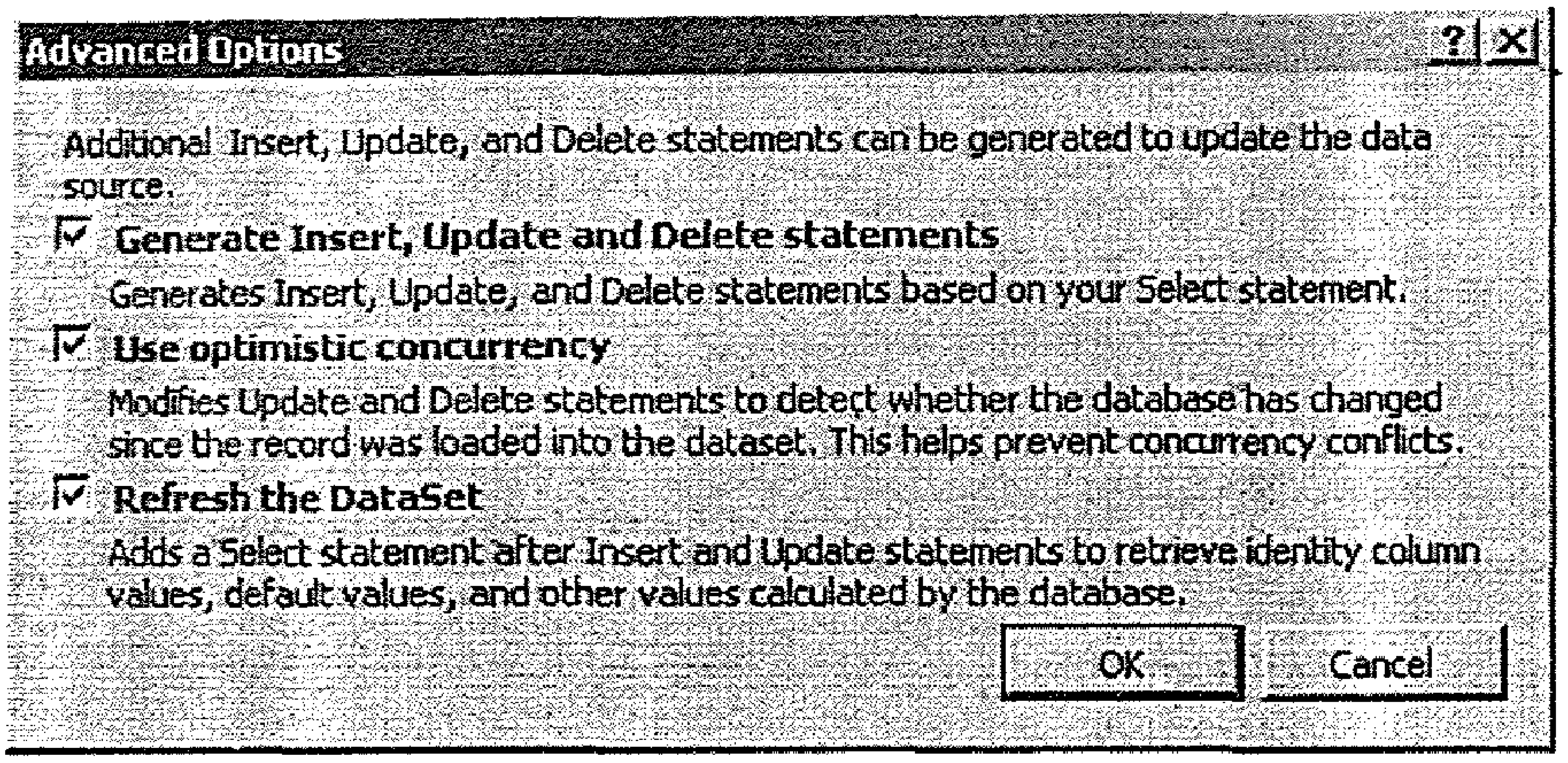
لكن لو كنت مصرا على وضع أكثر من جملة استعلام في موصل البيانات، ليقوم بملء مجموعة البيانات بأكثر من جدول، فاتبع الخطوات التالية:

- حدد موصل البيانات في صينية مكونات النموذج، واضغط F4 لعرض نافذة الخصائص.
- حدد الخاصية SelectCommand في نافذة الخصائص، واضغط العلامة + المجاورة لها لإسدال خصائص كائن الأمر.
- حدد الخاصية CommandText، واكتب في قيمتها جملة الاستعلام المكونة من أكثر من أمر.
- اضغط زر الحفظ أو انتقل إلى أي خاصية أخرى أو إلى النموذج.. ستظهر رسالة تسألك إن كنت تريد تحديث معاملات هذا الأمر.. اضغط زر الموافقة.

الآن سيكون كل شيء على ما يرام، وسيملأ موصل البيانات مجموعة البيانات بجدولي المؤلفين والكتب!

الطريف أنك لو أعدت فتح المعالج السحري فستجد جملة الاستعلام المركبة من أكثر من أمر معروضة في مربع النص، ولكنك ستظل تحصل على خطأ لو حاولت الانتقال إلى الخطوة التالية!

وتحتوي هذه النافذة أيضا على الزر "خيارات متقدّمة" Advanced Options، ولو ضغطته فستظهر لك نافذة بها الخيارات التالية:




١. **Generate Insert, Update And Delete Statements:**

استخدم هذا الخيار إذا كان برنامجك سيجري تعديلات في السجلات التي يحملها من قاعدة البيانات.. في هذه الحالة سيتمّ إنتاج جمل التحديث والإدراج والحذف آليا، بالاعتماد على أسماء الجداول والأعمدة الموجودة في جملة Select التي أنشأتها.

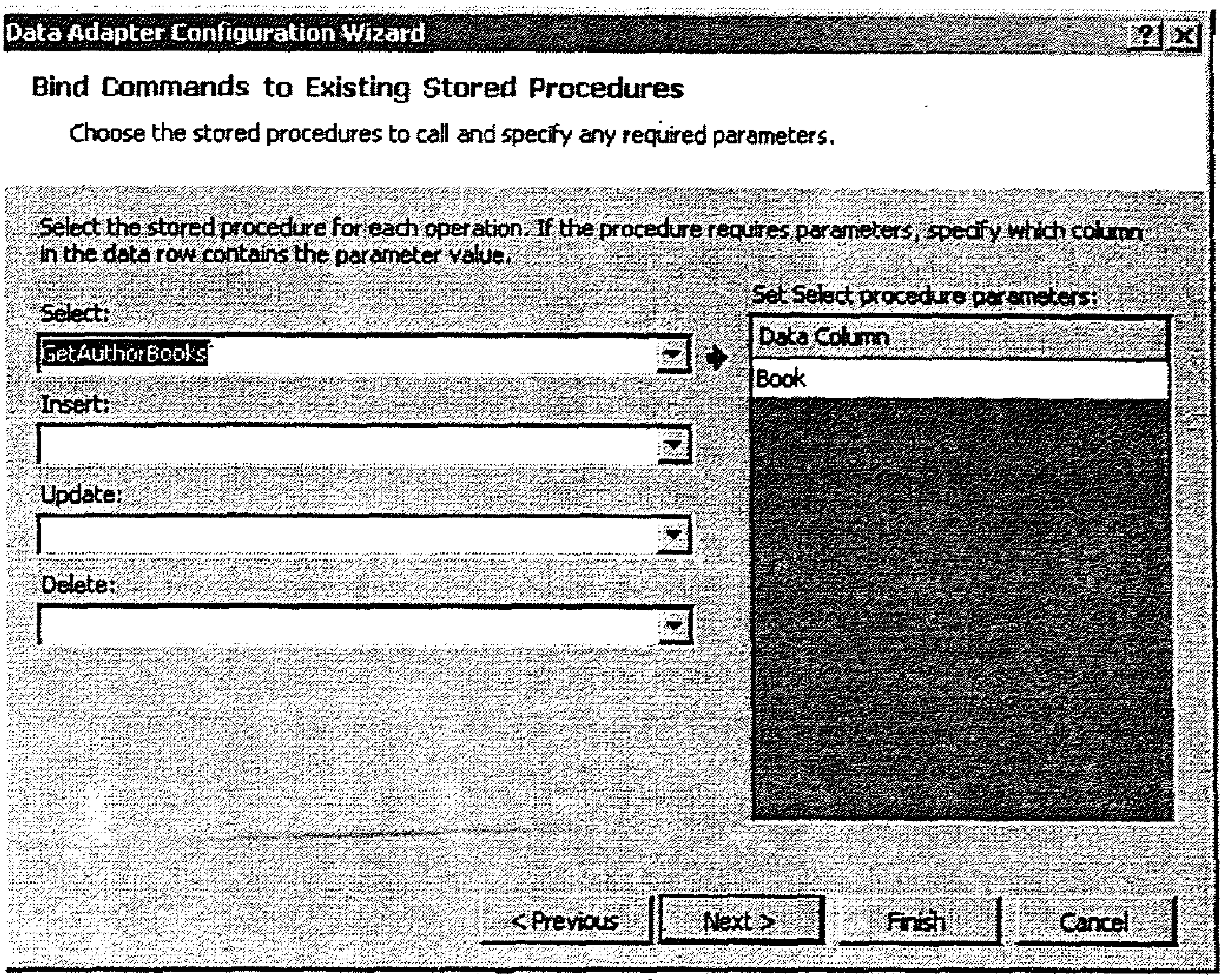
٢. **استخدم التطابق المتفائل Use Optimistic Concurrency:**

هذا الخيار يتيح لك استخدام التطابق المتفائل عند تحديث البيانات.. لاحظ أن إزالة علامة الاختيار يعني أنك تريد استخدام التطابق المتشائم.. هذا يؤثر فقط على صيغة جملة التحديث UpdateCommand، لكنه لن يكتب لك الكود المناسب لإغلاق سجلات قاعدة البيانات Lock، وعليك أن تكتب هذا الكود بنفسك من خلال كائن التعاملات Transaction Object.

٣. **إنعاش مجموعة البيانات Refresh The DataSet:**

هذا الخيار يضيف جملة تحديد SELECT بعد جملتي الإدراج والتحديث، وذلك لإنعاش مجموعة البيانات بعد تنفيذ أوامر الإدراج والتحديث، كما شرحنا سابقا.. لاحظ أن هذا الخيار غير متاح في قواعد بيانات Access، وذلك لأنه لا يسمح أصلا بتنفيذ أكثر من استعلام في المرة الواحدة.

- أما لو اخترت استخدام إجراءات مخزنة موجودة سابقا في قاعدة البيانات، فستظهر لك النافذة التالية:




هذه النافذة بها أربع قوائم منسدلة، تعرض كل منها أسماء الإجراءات المخزنة الموجودة في قاعدة البيانات، لتختار منها إجراء للتحديد Select والإدراج Insert والتحديث Update والحذف Delete.

ويوجد على يمين النافذة جدول يعرض بعض التفاصيل الخاصة بالإجراء المخزن.. فبالنسبة للإجراء الخاص بالتحديد، يعرض الجدول أسماء الأعمدة التي يعيد الإجراء المخزن محتوياتها (على سبيل المثال: الإجراء GetAuthorBooks يعيد العمود Book، الذي يحتوي على أسماء الكتب الخاصة بالمؤلف المطلوب.

أما بالنسبة لإجراءات الإدراج والتحديث والحذف، فيعرض هذا الجدول عمودين، أولهما يحتوي على المعاملات المعرفة في الإجراء المخزن، والثاني يحتوي على أسماء حقول مجموعة البيانات، التي سيتم ملء هذه المعاملات من قيمها لإرسالها إلى الإجراء المخزن.. وتحتوي كل خانة في هذا العمود على قائمة منسدلة يمكنك اختيار اسم الحقل منها.

وبعد الانتهاء من أي من النافذتين السابقتين، سيؤدي ضغط الزر Next إلى ظهور نافذة تعرض ملخّصا لخياراتك.. اضغط الزر Finish لإنهاء المعالج السحري وتنفيذ هذه الاختيارات، أو اضغط Cancel لإلغاء العملية.

لاحظ أنك تستطيع في كل نافذة من نوافذ هذا المعالج، الرجوع إلى النافذة السابقة بضغط الزر Previous.

بعد انتهاء المعالج ستجد أن خصائص موصل البيانات قد تم ضبطها لتوافق اختياراتك، كما ستجد كائن اتصال SqlConnection اسمه الافتراضي SqlConnection1 قد أضيف إلى صينية مكونات النموذج، ليستخدمه موصل البيانات في الاتصال بقاعدة البيانات.

# فئة باني أوامر قاعدة البيانات
# DbCommandBuilder Class

هذه الفئة هذه الفئة أساسية مجردة Abstract Base Class، تجب وراثتها MustInherit، وهي ترث فئة المكون Component Class، ويمكنك استخدامها لبناء أوامر التحديث والإدراج والحذف اللازمة لنقل التغييرات من أحد جداول مجموعة البيانات DataSet إلى قاعدة البيانات.. ولكي تفعل هذه الفئة هذا، عليك ربطها بموصل البيانات، بشرط أن يحتوي موصل البيانات على أمر التحديد SELECT Command.. وتقوم هذه الفئة بالاستجابة للحدث RowUpdating الخاص بموصل البيانات، حيث تستخلص المعلومات الأساسية عن تركيب الصف من أمر التحديد SELECT Command (كاسم الجدول وأسماء الأعمدة)، وتبني الأمر المناسب لتحديث أو حذف أو إدراج السجل الذي أطلق الحدث.
ويتم ربط موصل بيانات واحد فقط بباني أوامر واحد فقط.

وتمتلك هذه الفئة الخصائص التالية:

## موصل البيانات DataAdapter:

تقرأ أو تغير موصل البيانات الذي سيتم إنشاء أوامره.. لاحظ أن موصل البيانات يجب أن يحقق الشروط التالية:

١- أن يحتوي على أمر تحديث SELECT Command.
٢- أن يكون ضمن أعمدة النتيجة التي يعيدا أمر التحديد المفتاح الأساسي أو عمود متفرد Unique يميز كل صف.
٣- أن يعيد أمر التحديد النتائج من جدول واحد فقط.. استعلامات الربط بين أكثر من جدول مرفوضة.

وعند الإخلال بأي من هذه الشروط، سيحدث خطأ ويرفض باني الأوامر إنتاج أوامر التحديث والحذف والإدراج.. ويمكنك استدعاء الوسيلة Dispose الخاصة بباني الأوامر، لإنهاء ارتباطه بموصل البيانات، والوقوف عن استخدام الأوامر التي أنشأها.

### وضع جميع القيم SetAllValues:

إذا جعلت قيمتها True، ينتج باني الأوامر أمر تحديث Update يحدث جميع قيم السجل.. أما إذا جعلتها False، فسينتج أمر تحديث يحدث فقط قيم الحقول التي تغيرت في مجموعة البيانات.. لاحظ أن باني الأوامر يتابع الحدث RowUpdating الخاص بموصل البيانات، يفحص كل صف قبل تحديثه، ومن ثم ينتج استعلام التحديث المناسب لهذا الصف تبعا للتغييرات التي حدثت فيه في مجموعة البيانات.

طبعا هذا أذكى وأسرع وأقل عبئا على خادم سيكويل، لكنه قد يؤدي إلى نتائج غير متوقعة إذا كنت تحل مشاكل التوافق Concurrency Conflicts باستخدام طريقة "الأخير يكسب"، بحيث تحفظ تغييراتك مباشرة، فهذا قد يجعل قيم السجل تتكون من مزيج من تغييراتك وتغييرات مستخدم آخر.. أما إذا كنت تستخدم طابع الوقت أو تقارن كل الحقول، فلا خوف من هذه المشكلة.

### خيار التعارض ConflictOption:

تحدد كيف يتم إنشاء أمر التحديث UPDATE والحذف DELETE لتلافي مشاكل التطابق Concurrency Conflicts.. وتأخذ هذه الخاصية إحدى قيم المرقم ConflictOption التالية:

| | |
|---|---|
| CompareAll SearchableValues | يتم البحث عن السجل المراد تحديثه في قاعدة البيانات، بمقارنة جميع قيم الحقول العددية والنصية الصغيرة، للتأكد من أن السجل لم تدخل عليه أية تعديلات. |
| Compare RowVersion | يتم البحث عن السجل المراد تحديثه في قاعدة البيانات، بمقارنة حقل الإصدار.. هذا يتطلب وجود حقل من النوع TimeStamp في الجدول. |
| Overwrite Changes | يتم البحث عن السجل المراد تحديثه في قاعدة البيانات، باستخدام مفتاحه الأساسي فقط، وهذا يعني أن التغييرات الخاصة ببرنامجك سيتم حفظها في السجل لتلغي أية تغييرات أدخلها المستخدمون الآخرون. |

### قوس الفتح QuotePrefix:
تقرأ أو تغير النص المستخدم كقوس فتح، لاستخدامه مع أسماء الجداول والأعمدة التي تحتوي على مسافات أو حروف غير مقبولة.

### قوس الإغلاق QuoteSuffix:
تقرأ أو تغير النص المستخدم كقوس إغلاق.
مع قواعد البيانات المألوفة، يكون قوسا الفتح والإغلاق [ ].

### موضع الفهرس CatalogLocation:
تقرأ أو تغير الموضع الذي سيوضع فيه اسم قاعدة البيانات، عند تكوين المسارات الكاملة لأسماء الجداول في أوامر SQL.. وهي تأخذ إحدى قيمتي المرقم CatalogLocation التاليتين:

| | |
|---|---|
| Start | يوضع اسم قاعدة البيانات في بداية المسار. |
| End | يوضع اسم قاعدة البيانات في نهاية المسار. |

### فاصل الفهرس CatalogSeparator:
أو تغير النص المستخدم كفاصل بين اسم قاعدة البيانات واسم الجدول عند كتابة المسار الكامل.. المألوف أن تستخدم النقطة . كفاصل، مثل Books.Author .

### فاصل المخطط SchemaSeparator:
تقرأ أو تغير النص المستخدم كفاصل بين اسم المخطط Schema واسم معرف Identifier موجود في هذا المخطط.. والمألوف أن تستخدم النقطتان المتعامدتان : كفاصل، مثل: Person:CustomerName.

كما تملك هذه الفئة الوسائل التالية:

### إنعاش المخطط RefreshSchema:
تحذف أوامر التحديث والحذف والإدراج التي قام باني الأوامر ببنائها.. هذا ضروري إذا أردت أن تجعل باني الأوامر ينعش معلوماته عن موصل البيانات، حيث إن باني الأوامر يبني أوامره بعد أول عملية تحديث لقاعدة البيانات، ويستخدمها كما هي بعد هذا.. لهذا عليك استدعاء هذه الوسيلة إذا

قمت بتغيير استعلام التحديد أو وقت الانتظار CommandTimeout أو كائن التعاملات Transaction الذي يستخدمه أمر التحديد، لكي يعيد باني الأوامر إنشاء أوامر التحديث والحذف والإدراج لتلائم هذه التغييرات.

**معرفة أمر التحديث GetUpdateCommand:**

تعيد كائن الأمر DbCommand الذي تم إنتاجه لتحديث قاعدة البيانات.

وهناك صيغة ثانية لهذه الوسيلة، تستقبل معاملا منطقيا، إذا جعلته False، فسيستخدم باني الأوامر في أمر التحديث، معاملات لها الأسماء P1 و P2 و P3 وهكذا... (مثل SET Author = @P1).. وهذه هي الحالة الافتراضية في الصيغة الأولى لهذه الوسيلة.

أما إذا جعلت قيمة هذا المعامل True، فسيتم إنتاج معاملات لها نفس أسماء الأعمدة ، كلما كان هذا ممكنا (مثل SET Author = @Author).. لاحظ أن محاولة إنتاج هذه الأسماء ستسبب خطأ إلا إذا جعلت كائن DbMetaDataColumnNames يلتزم بالشروط التالية:

١- تحديد أقصى طول ممكن لأسماء المعاملات، من خلال الخاصية ParameterNameMaxLength.

٢- توضيح صيغة أسماء المعاملات، من خلال الخاصية ParameterNamePattern.

٣- تحديد تنسيق العلامة المميزة للمعامل، من خلال الخاصية ParameterMarkerFormat.

**معرفة أمر الإدراج GetInsertCommand:**

تعيد كائن الأمر DbCommand الذي تم إنتاجه لإدراج صف جديد في قاعدة البيانات.. وهي مماثلة في صيغتيها للوسيلة GetUpdateCommand.

**معرفة أمر الحذف GetDeleteCommand:**

تعيد كائن الأمر DbCommand الذي تم إنتاجه لحذف صف من قاعدة البيانات.. وهي مماثلة في صيغتيها للوسيلة GetUpdateCommand.

**تقويس المعرف QuoteIdentifier:**
أرسل إلى هذه الوسيلة نصا يمثل مسارا كاملا لأحد عناصر قاعدة البيانات (مثل Books.Author.ID)، لتعيد إليك نفس المسار بعد وضع كل أسماء العناصر الموجودة فيه بين قوسين (مثل [Books].[Author].[ID]).. لاحظ أن المعرف المحاط بقوسين فعلا سيتم تجاهله.

**إزالة تقويس المعرف UnquoteIdentifier:**
أرسل إلى هذه الوسيلة نصا يمثل مسارا مقوسا لأحد عناصر قاعدة البيانات (مثل [Books].[Author].[ID])، لتعيد إليك نفس المسار بعد إزالة جميع الأقواس منه (مثل Books.Author.ID).

والفئات التالية ترث الفئة DbCommandBuilder:
١. OdbcCommandBuilder Class.
٢. OleDbCommandBuilder Class.
٣. OracleCommandBuilder Class.
٤. SqlCommandBuilder Class.

وسنتعرف هنا فقط على الفئة SqlCommandBuilder.

## فئة باني أوامر سيكويل
## SqlCommandBuilder Class

هذه الفئة ترث الفئة DbCommandBuilder، وهي تمتلك نفس خصائصها ووسائلها، مع فارق بسيط أنها مخصصة للتعامل مع سيكويل سيرفر وأوامره SqlCommand.

ولحدث إنشاء هذه الفئة صيغتان:

١- الأولى بدون معاملات.

٢- والثانية تستقبل موصل البيانات SqlDataAdapter الذي سيرتبط به باني الأوامر.

وتمتلك هذه الفئة الوسيلة الجديدة التالية:

### اشتقاق المعاملات DeriveParameters:

أرسل إلى هذه الوسيلة كائن أمر SqlCommand مجهز لتنفيذ إجراء مخزن، لتقوم هذه الوسيلة بالاتصال بقاعدة البيانات، والحصول على معلومات عن معاملات الإجراء المخزن، واستخدامها لإضافة المعاملات المناسبة غلى مجموعة المعاملات Parameters Collection الخاصة بكائن الأمر.. لاحظ أن خطأ سيحدث لو أرسلت إلى هذه الوسيلة كائن أمر يتعامل مع استعلام SQL أو يحتوي على اسم إجراء مخزن غير صحيح. ويعيب هذه الوسيلة أنها تحتاج إلى الاتصال بقاعدة البيانات مرة إضافية لإحضار بيانات المعاملات، لأنك بالتأكيد ستتصل مرة ثانية لتنفيذ الأمر.

والمشروع CommandBuilder يريك مثالا على استخدام هذه الفئة لإنتاج أوامر التحديث والإدراج والحذف.

## خرائط البيانات Data Mapping:

يتيح لك موصل البيانات عمل خراط للجداول Table Mapping، وذلك بإعادة تسمية الجداول والأعمدة بأسماء خاصة بك، وربطها بالأسماء الحقيقية في قاعدة البيانات.. هذا يحقق لك الفوائد التالية:

١- تغيير أسماء الجداول أو الأعمدة في مجموعة البيانات كما يناسب برنامجك، دون العبث بقاعدة البيانات الأصلية.

٢- عرض أسماء الأعمدة للمستخدم بالطريقة التي تناسبك.

٣- إذا كان لديك جدولان لهما نفس الاسم في قاعدتي بيانات مختلفتين، فإن بإمكانك إضافتهما إلى نفس مجموعة البيانات، وذلك بتغيير اسميهما من خلال خريطة الجدول الخاصة بكل منهما.

وتنقسم خرائط البيانات إلى نوعين:

### ١- خريطة الجدول Table Mapping:

حيث تكون لكل جدول خريطة، يذكر فيها اسمه الأصلي في قاعدة البيانات واسمه الجديد في مجموعة البيانات.. وتوضع خرائط الجداول في المجموعة TableMappings في موصل البيانات.

وهناك نقطة هامة يجب أن تنتبه إليها عند إنشاء هذه الخرائط، هي ان موصل البيانات يعطي أسماء افتراضية للجداول، تختلف عن أسمائها الأصلية (مثل Table و Table1... إلخ).. هذا قد يسبب لك ارتباكا وأنت تنشئ خرائط الجداول، فستبدو لك وظيفتها عكسية، فبدلا من أن تعطي الجدول الأصلي اسما جديدا، ستحاول أن تعيد تسمية الاسم الافتراضي الخاص بموصل البيانات باسم الجدول الأصلي!

لكن بقليل من التأمل، ستفهم لماذا يفعل موصل البيانات هذا.. فاستعلام التحديد في معظم الحالات لا يعيد جدولا من قاعدة البيانات، بل قد يعيد أجزاء من عدة جداول (كما في حالة الربط Joining) أو قد يعيد نتائج محسوبة من جدول أو أكثر (كما في حالة التجميع Aggregation).. لهذا يريح موصل البيانات نفسه من كل هذه الاحتمالات المعقدة، ويسمي الجداول الناتجة من الاستعلام بأسماء افتراضية، ويترك لك حرية إنشاء خريطة الجداول التي تصحح فيها الأسماء بطريقتك.

والمشروع Mapping يريك مثالا على هذا.. فنحن نستخدم استعلام ربط يعيد المؤلفين وكتبهم.. نتيجة هذا الاستعلام ستحتوي على جدول مخلق، سيعطيه موصل البيانات الاسم الافتراضي Table، لهذا استخدمنا خريطة الجدول لإعادة تسميته Authors-Books.

٢- خريطة العمود Column Mapping:

حيث تكون لكل عمود خريطة، يذكر فيها اسمه الأصلي في قاعدة البيانات واسمه الجديد في مجموعة البيانات.. وتوضع خرائط الأعمدة في المجموعة ColumnMappings في خريطة الجدول الذي تنتمي إليه.

ولا يحتاج موصل البيانات إلى تسمية الأعمدة بأسماء افتراضية، لسبب بسيط: هو أن كل عمود يتم ذكره صراحة في استعلام التحديد، وحتى الأعمدة المولدة (الأعمدة المحسوبة) يتم تسميتها إجباريا باستخدام الفقرة As، لهذا فإن موصل البيانات يعرف يقينا اسم كل عمود في النتيجة.. ولا يتدخل موصل البيانات لإعادة تسمية العمود، إلا في حالة وجود عمودين بنفس الاسم (يمكن أن يحدث هذا لو كنت تستخدم أكثر من جملة SELECT في أمر التحديد مثلا).

وأهم استخدام لخرائط الأعمدة، هو إعادة تسمية الأعمدة بطريقة تصلح لعرضها للمستخدم.. والمشروع Mapping يريك مثالا على هذا، حيث استخدمنا خريطة الأعمدة لإعادة تسمية العمود Author بالاسم "المؤلف"، والعمود Books بالاسم "الكتاب".. هذان الاسمان سيظهران في جدول العرض وهذا مناسب للمستخدم العربي للبرنامج.

لاحظ أنك بعد عمل خرائط الربط، ستستخدم اسم الجدول الجديد واسمي العمودين العربيين في الكود عند التعامل معهما من خلال مجموعة البيانات.. مثلا:

```
Dim T = Ds.Tables("Authors-Books")
MsgBox(T.Columns("المؤلف").MaxLength)
```

في الحقيقة هناك حل آخر لعرض أسماء الأعمدة بأسماء عربية دون استخدام خريطة الأعمدة، وذلك باستخدام خصائص جدول العرض نفسه لإعادة تسمية عنوان العمود، كما سنرى فيها بعد.

والآن، فلنتعرف على خرائط الجداول والأعمدة، وكيفية استخدامها.. وسنتفق هنا على استخدام التعبير "اسم الجدول الأصلي" للإشارة إلى الاسم الذي يمنحه موصل البيانات للجدول، مع ملاحظة أن هذا الاسم حساس لحالة الأحرف.. كما سنستخدم التعبير "اسم العمود الأصلي" للإشارة إلى اسم العمود في قاعدة البيانات، أو الاسم البديل الذي سماه به موصلا البيانات إن حدث تعارض بين عمودين، مع ملاحظة أن هذا الاسم غير حساس لحالة الأحرف.

## واجهة مجموعة خرائط الجداول
## ITableMappingCollection Interface

هذه الواجهة ترث واجهة القائمة IList، وهي قائمة تحتوي على خرائط الجداول. وإضافة إلى ما ترثه من واجهة القائمة من عناصر، تمتلك الوسيلة الوحيدة التالية:

- **معرفة الخريطة من جدول مجموعة البيانات GetByDataSetTable:**
أرسل إلى هذه الوسيلة اسم جدول موجود في مجموعة البيانات DataSet، لتعيد إليك كائنا تمثل الواجهة ITableMapping، يحتوي على خريطة هذا الجدول.. ومن المتوقع أن يكون هذا الكائن من نوع الفئة DataTableMapping التي سنتعرف عليها لاحقا.

كما تضيف هذه الواجهة صيغة أخرى لبعض عناصر القائمة التقليدية، مثل:

- **العنصر Item:**
تستقبل الصيغة الثانية لهذه الخاصية اسم الجدول الأصلي (وهو حساس لحالة الأحرف)، وتعيد كائنا Object يحتوي على خريطة هذا الجدول إن وجدت في القائمة، وإن لم توجد فسيحدث خطأ.
كما يمكنك استخدام هذه الخاصية لتغيير كائن خريطة الجدول، فهي قابلة للقراءة وللكتابة أيضا.

- **إضافة Add:**
تستقبل الصيغة الثانية لهذه الوسيلة معاملين نصيين Strings، أولها هو اسم الجدول الأصلي (وهو حساس لحالة الأحرف Case-Sensitive)، وثانيهما هو اسم الجدول في مجموعة البيانات.. وتقوم هذه الوسيلة بإنشاء كائن خريطة جدول DataTableMapping يمثل العلاقة بين الجدولين وتضيفه إلى القائمة، وتعيد نسخة من الواجهة ITableMapping تشير إلى هذا الكائن.

- **تحتوي على Contains:**
تستقبل الصيغة الثانية لهذه الوسيلة اسم الجدول الأصلي، وتعيد True إذا كانت هناك خريطة لهذا الجدول في القائمة.

- **رقم العنصر IndexOf:**
تستقبل الصيغة الثانية لهذه الوسيلة اسم الجدول الأصلي، وتعيد رقم الخانة التي يوجد بها كائن خريطة هذا الجدول في القائمة إن وجد، أو تعيد -١ إن لم توجد خريطة لهذا الجدول.

- **حذف من موضع RemoveAt:**
تستقبل الصيغة الثانية لهذه الوسيلة اسم الجدول الأصلي، وتبحث في القائمة عن كائن خريطة هذا الجدول، وتحذفه إن وجدته.

| **ملحوظة:** |
| --- |
| هذه الصيغة تبدو مختلفة في وظيفتها عن الصيغة الأولى المألوفة، التي تستقبل رقم خانة في القائمة وتحذفها لإزالة خريطة الجدول الموجودة بها من القائمة.. وإن شئت رأيي، كان المنطقي أن تكون هذه الصيغة الجديدة هي الصيغة الثانية للوسيلة Remove وليس RemoveAt منعا للالتباس!! |

## فئة مجموعة خرائط الجداول
## DataTableMappingCollection Class

هذه الفئة تمثل الواجهة ITableMappingCollection، وهي تعمل كقائمة، عناصرها من نوع الفئة DataTableMapping التي سنتعرف عليها بعد قليل.
وإضافة إلى ما ترثه من الواجهة ITableMappingCollection والواجهة IList من خصائص ووسائل، تمتلك هذه المجموعة الوسيلتين الجديدتين التاليتين:

### معرفة خريطة الجدول GetTableMappingBySchemaAction:

تبحث في مجموعة الخرائط عن الخريطة التي تربط بين اسم الجدول الأصلي واسم الجدول في مجموعة البيانات، فإن وجدته تعيد كائنا من النوع DataTableMapping يمثل هذه الخريطة.. وتستقبل هذه الوسيلة المعاملات التالية بالترتيب:

- مجموعة خرائط الجداول DataTableMappingCollection التي سيتم البحث فيها.
- اسم الجدول الأصلي.
- اسم الجدول في مجموعة البيانات.
- إحدى قيم المرقم MissingMappingAction التي تحدد ماذا سيحدث إذا لم تكن خريطة الجدول موجودة، كما هو موضح في الجدول التالي:

| | |
|---|---|
| Passthrough | يتم إنشاء خريطة جدول جديدة، تحمل اسم الجدول الأصلي المرسل في المعامل الأول، واسم جدول مجموعة البيانات المرسل في المعامل الثاني. |
| Ignore | يتم تجاهل الخطأ، وتعيد الوسيلة Nothing. |
| Error | يتم إطلاق خطأ من النوع InvalidOperationException. |

### رقم جدول مجموعة البيانات IndexOfDataSetTable:

أرسل إلى هذه الوسيلة اسم الجدول في مجموعة البيانات، لتعيد إليك رقم خريطة الجدول في مجموعة الخرائط الحالية.. وتعيد هذه الوسيلة -١ إذا لم تعثر على خريطة هذا الجدول.

## واجهة خريطة الجدول
## ITableMapping Interface

تمتلك هذه الواجهة الخصائص اللازمة لرسم خريطة الربط بين الجدول الأصلي والجدول الموجود في مجموعة البيانات.. وهذه الخصائص هي:

**جدول المصدر SourceTable:**
تقرأ أو تغير اسم الجدول الأصلي.

**جدول مجموعة البيانات DataSetTable:**
تقرأ أو تغير اسم الجدول في مجموعة البيانات.

**خرائط الأعمدة ColumnMappings:**
تعيد كائنا يمثل واجهة خرائط الأعمدة IColumnMappingCollection، وهو تحديدا من نوع الفئة ColumnMappingCollection، التي يمكنك أن تضيف إليها خرائط الربط بين الأعمدة.

## فئة خريطة الجدول DataTableMapping Class

هذه الفئة تمثل الواجهة DataTableMapping، وهي تحتوي على المعلومات اللازمة لربط الجدول في مجموعة البيانات بالجدول الأصلي.
ولحدث إنشاء هذه الفئة ثلاث صيغ:

١- الصيغة الأولى بدون معاملات.
٢- والصيغة الثانية تستقبل اسم الجدول الأصلي واسم الجدول في مجموعة البيانات.
٣- والصيغة الثالثة تزيد على الصيغة السابقة بمعامل ثالث، يستقبل مصفوفة من النوع DataColumnMapping، تحتوي على معلومات الربط بين أعمدة الجدولين.

وتمتلك هذه الفئة هذه الوسائل الجديدة:

### معرفة عمود البيانات GetDataColumn:

تعيد كائن عمود البيانات DataColumn الذي يمثل العمود المحدد بالمعاملات المرسلة، وهي بالترتيب:

- اسم العمود الأصلي.
- كائن النوع Type الذي يمثل نوع هذا العمود.
- كائن الجدول DataTable الذي يمثل الجدول في مجموعة البيانات.
- إحدى قيم المرقم MissingMappingAction التي تحدد ماذا سيحدث لو لم يتم العثور على العمود في خريطة الجدول.. وقد تعرفنا على قيم هذا المرقم سابقا.
- إحدى قيم المرقم MissingSchemaAction التي تحدد ماذا سيحدث لو لم يتم العثور على العمود في مخطط الجدول Schema، وهذه القيم هي:

| | |
|---|---|
| Add | يضاف العمود إلى مخطط الجدول. |
| AddWithKey | يضاف العمود والمفتاح الأساسي Primary Key إلى مخطط الجدول. |
| Ignore | يتم تجاهل العمود. |
| Error | يتم إطلاق خطأ من النوع InvalidOperationException. |

- **معرفة الجدول GetDataTableBySchemaAction:**

تعيد كائن جدول البيانات DataTable الموجود في مجموعة البيانات، والمذكور اسمه في خريطة الربط.. وتستقبل هذه الوسيلة معاملين:

- كائن مجموعة البيانات DataSet.
- إحدى قيم المرقم MissingSchemaAction، التي توضح التصرف المناسب إذا لم يتم العثور على هذا الجدول في مجموعة البيانات.

- **معرفة خريطة العمود GetColumnMappingBySchemaAction:**

تعيد كائن خريطة العمود DataColumnMapping للعمود الذي تريده، وهي تستقبل معاملين:

- اسم العمود الأصلي.
- إحدى قيم المرقم MissingMappingAction الذي تعرفنا عليه من قبل، والتي توضح رد الفعل إذا لم يتم العثور على هذا العمود.

## واجهة مجموعة خريطة العمود

## IColumnMappingCollection Interface

هذه الواجهة ترث واجهة القائمة IList، وهي تملك وسيلة وحيدة جديدة، وهي:

- **معرفة الخريطة بواسطة عمود مجموعة البيانات GetByDataSetColumn:**
أرسل إلى هذه الوسيلة اسم العمود في مجموعة البيانات، لتعيد إليك كائنا يمثل واجهة خريطة العمود IColumnMapping الذي يحتوي على معلومات ربط هذا العمود بالعمود الأصلي.. وسيكون هذا الكائن من نوع الفئة DataColumnMapping تحديدا.

وتضيف هذه الواجهة صيغة أخرى لبعض خصائص ووسائل القائمة التقليدية، مثل:

- **العنصر Item:**
تستقبل الصيغة الثانية لهذه الخاصية اسم العمود الأصلي (وهو حساس لحالة الأحرف Case-Sensitive)، وتعيد كائنا Object يحتوي على خريطة هذا العمود إن وجدت في القائمة، وإن لم توجد فسيحدث خطأ.

- **إضافة Add:**
تستقبل الصيغة الثانية لهذه الوسيلة معاملين نصيين Strings، أولها هو اسم العمود الأصلي، وثانيهما هو اسم العمود في مجموعة البيانات.. وتقوم هذه الوسيلة بإنشاء كائن خريطة أعمدة DataColumnMapping يمثل العلاقة بين العمودين وتضيفه إلى القائمة، وتعيد نسخة من الواجهة IColumnMapping تشير إلى هذا الكائن.

- **تحتوي على Contains:**
تستقبل الصيغة الثانية لهذه الوسيلة اسم العمود الأصلي (وهو حساس لحالة الأحرف)، وتعيد True إذا كانت هناك خريطة لهذا العمود في القائمة.

- **رقم العنصر IndexOf:**

تستقبل الصيغة الثانية لهذه الوسيلة اسم العمود الأصلي، وتعيد رقم الخانة التي يوجد بها كائن خريطة هذا العمود في القائمة إن وجد، أو تعيد -١ إن لم توجد خريطة لهذا العمود.

- **حذف من موضع RemoveAt:**

تستقبل الصيغة الثانية لهذه الوسيلة اسم العمود الأصلي، وتبحث في القائمة عن كائن خريطة هذا العمود، وتحذفه إن وجدته.

## فئة مجموعة خريطة العمود
## DataColumnMappingCollection Class

هذه الفئة تمثل الواجهة IColumnMappingCollection، وهي تعمل كقائمة تحتوي على كائنات من نوع الفئة DataColumnMapping، التي ترسم خرائط الربط بين أعمدة مجموعة البيانات والأعمدة الأصلية.
وإضافة إلى ما تمثله من خصائص ووسائل الواجهة IColumnMappingCollection، تمتلك هذه الفئة الوسائل التالية:

### معرفة عمود البيانات GetDataColumn:
مماثلة للوسيلة GetDataColumn الخاصة بالفئة DataTableMapping، مع فارق وحيد، هو أنها هنا وسيلة مشتركة Shared، لهذا تمتلك معاملا زائدا، هو المعامل الأول الذي يستقبل مجموعة خرائط الأعمدة DataColumnMappingCollection التي سيتم البحث فيها.

### معرفة خريطة العمود GetColumnMappingBySchemaAction:
مماثلة للوسيلة GetColumnMappingBySchemaAction الخاصة بالفئة DataTableMapping، مع فارق وحيد، هو أنها هنا وسيلة مشتركة Shared، لهذا تمتلك معاملا زائدا، هو المعامل الأول الذي يستقبل مجموعة خرائط الأعمدة DataColumnMappingCollection التي سيتم البحث فيها.

### معرفة رقم العمود IndexOfDataSetColumn:
تستقبل اسم العمود في مجموعة البيانات، وتعيد رقم الخانة التي يوجد بها كائن خريطة هذا العمود في القائمة إن وجد، أو تعيد -١ إن لم توجد خريطة لهذا العمود.

## واجهة خريطة العمود
## IColumnMapping Interface

تمتلك هذه الواجهة خاصيتين فقط، تستخدمان لربط عمود من مجموعة البيانـــات، بالعمود الأصلي، وهما:

- **عمود المصدر SourceColumn:**
تحدد اسم العمود الأصلي.

- **عمود مجموعة البيانات DataSetColumn:**
تحدد اسم العمود في مجموعة البيانات.

## فئة خريطة العمود

## DataColumnMapping Class

هذه الفئة تمثل الواجهة IColumnMapping والواجهة ICloneable، وهي ترسم خريطة الربط بين عمود مجموعة البيانات والعمود الأصلي.

ولحدث إنشاء هذه الفئة صيغتان:

١- الأولى بدون معاملات.

٢- والثانية لها معاملان نصيان، يستقبلان اسم العمود الأصلي واسم عمود مجموعة البيانات على الترتيب.

وإضافة إلى ما تمثله من خصائص، تمتلك هذه الفئة الوسيلة الوحيدة الجديدة التالية:

### معرفة عمود البيانات GetDataColumnBySchemaAction:

تعيد كائن عمود البيانات DataColumn المطلوب تبعا للمعاملات التالية:

- كائن جدول البيانات DataTable الذي يحتوي العمود.. لاحظ أن اسم العمود في الجدول تحدده الخاصية DataSetColumn الخاصة بخريطة العمود الحالية.
- كائن النوع Type الذي يمثل نوع بيانات العمود.
- إحدى قيم المرقم MissingSchemaAction، تحدد ماذا سيحدث إن لم يوجد العمود في مخطط الجدول.

وتوجد صيغة أخرى لهذه الوسيلة، وهي صيغة مشتركة Shared، لهذا تزيد بمعاملين على الصيغة السابقة، هما المعامل الأول والثاني، اللذان يستقبلان اسم العمود الأصلي واسم العمود في مجموعة البيانات على الترتيب.

ولقد استخدمنا الكود التالي في المشروع Mapping لإعادة تسمية الجدول وعموديه:

```
Dim TM = DaAuthors.TableMappings.Add(
            "Table", "Authors-Books")
TM.ColumnMappings.Add("Author", "المؤلف")
TM.ColumnMappings.Add("Book", "الكتاب")
```

# مصانع المزودات Provider Factories

لعلك شعرت بالاستياء من وجود أكثر من نوع من نفس الكائن للتعامل مع مزودات قواعد البيانات المختلفة، مثل:

- كائنات الاتصال مثل OleDbConnection و SqlConnection و OracleConnection.
- كائنات الأمر مثل OleDbCommand و SqlCommand و OracleCommand.
- كائنات قراءة البيانات مثل OleDbDataReader و SqlDataReader و OracleDataReader.
- موصلات البيانات مثل OleDbDataAdapter و SqlDataAdapter و OracleDataAdapter.

فهذا يجعلك تكتب كودا مختلفا لكل نوع من أنواع قواعد البيانات، رغم أن الاختلاف ينحصر فقط في جمل تعريف الكائنات، وليس في فكرة الكود!

ولقد قدمت دوت نت ٢٠٠٥ حلا لهذه المشكلة بإضافتين هامتين:

١- تعريف الفئات العامة في النطاق System.Data.Common، مثل:

- الفئة DbConnection التي تشتق منها جميع كائنات الاتصال.
- الفئة DbCommand التي تشتق منها جميع كائنات الأوامر.
- الفئة DbDataReader التي تشتق منها كل قارئات البيانات.
- الفئة DbDataAdapter التي تشتق منها كل موصلات البيانات.

هذا يجعل من الممكن استخدام الفئة الأم للتعامل مع أي نوع من أنواع الفئات المشتقة منها (راجع مفهوم الفئات الأساسية المجردة Abstract Base Classes وتعدد الأسماء Polymorphism في فصل الوراثة في كتاب "من الصفر إلى الاحتراف: فيجيوال بيزيك دوت نت").

٢- إضافة الفئتين DbProviderFactories و DbProviderFactory إلى النطاق System.Data.Common، لإمدادك بمصنع خاص بمزود البيانات الذي تريد التعامل معه، مما يكمل قدرتك على تعميم الكود، كما سنرى بعد قليل.

وسنتعلم في هذا الفصل كيف نستخدم هاتين الإمكانيتين لكتابة كود واحد للتعامل مع أنواع مختلفة من قواعد البيانات، وسنستخدمه للتعامل مع قاعدة بيانات الكتب في كل من سيكويل سيرفر وآكسيس.

## فئة مصانع المزودات DbProviderFactories Class

تعتبر هذه الفئة مجرد مدخل لاستخدام الفئة DbProviderFactory، وهي لا تمتلك إلا وسيلتين مشتركتين، هما:

### معرفة فئات المصانع GetFactoryClasses:

تعيد جدول بيانات DataTable، يحتوي على بيانات عن مصانع المزودات المتاحة على جهاز المستخدم.. ويمثل كل صف في هذا الجدول أحد المزودات، بينما تعرض الأعمدة تفاصيل هذا المزود.. وهذه الأعمدة هي:

| | |
|---|---|
| Name | اسم مزود البيانات. |
| Description | وصف مختصر لمزود البيانات. |
| InvariantName | الاسم الثابت للمزود، والذي يمكنك استخدامه للحصول على المصنع الخاص به. |
| Assembly QualifiedName | الاسم الكامل لمزود البيانات، وهو يحتوي على التفاصيل الكافية عنه، مثل الإصدار والثقافة التي يستخدمها. |

ويمكنك رؤية هذه التفاصيل بنفسك في المشروع DataProviders، فهو يعرض ناتج هذه الوسيلة في جدول عرض.

### معرفة المصنع GetFactory:

تعيد مصنع المزود DbProviderFactory الذي يتيح لك التعامل مع مزود معين.. ولهذه الوسيلة الصيغتان التاليتان:

١- الصيغة الأولى تستقبل الاسم الثابت للمزود InvariantName.

٢- والصيغة الثانية تستقبل صف البيانات DataRow الذي يحتوي على تفاصيل المزود.. ويمكنك الحصول على هذا الصف من الجدول العائد من الوسيلة GetFactoryClasses.

دعنا إذن نتعرف على الفئة DbProviderFactory.

## فئة مصنع المزود DbProviderFactory Class

هذه الفئة أساسية مجردة تجب وراثتها، وهي تمتلك العناصر اللازمة للتعامل مع مزود البيانات الذي خصصت للتعامل معه.
وتمتلك هذه الفئة الخاصية التالية:

**يمكنه إنشاء عداد لمصدر البيانات CanCreateDataSourceEnumerator:**

تعيد True إذا كان مصنع المزود يسمح باستخدام الفئة DbDataSourceEnumerator للمرور عبر كل خوادم البيانات المتاحة.. وسنتعرف على هذه الفئة بعد قليل.

كما تمتلك هذه الفئة الوسائل التالية:

**إنشاء باني نص الاتصال CreateConnectionStringBuilder:**

تعيد باني نص الاتصال من النوع العام DbConnectionStringBuilder، لكنه يكون مخصصا للتعامل مع مزود البيانات الذي أنشأت المصنع الحالي للتعامل معه.

**إنشاء كائن اتصال CreateConnection:**

تعيد كائن اتصال من النوع العام DbConnection، لكنه يكون مخصصا للتعامل مع مزود البيانات الذي أنشأت المصنع الحالي للتعامل معه.
لاحظ أن كائن الاتصال الذي ستحصل عليه غير مرتبط بأي نص اتصال، لهذا عليك وضع نص الاتصال في الخاصية ConnectionString الخاصة به قبل محاولة فتح الاتصال.

**إنشاء كائن أمر CreateCommand:**

تعيد كائن أمر من النوع العام DbCommand، لكنه يكون مخصصا للتعامل مع مزود البيانات الذي أنشأت المصنع الحالي للتعامل معه.. وأنت تعرف أنك تستطيع الحصول على قارئ البيانات من كائن الأمر باستدعاء الوسيلة ExecuteReader.
لاحظ أن كائن الأمر الذي ستحصل عليه ليس مرتبطا بأي اتصال، لهذا عليك ربطه بكائن الاتصال الذي حصلت عليه من الوسيلة

CreateConnection، وهو ما فعلناه في الدالة CreateCommand في المشروع Factories كالتالي:

```
Dim Command = Fac.CreateCommand
Command.Connection = Cn
```

ويمكنك أداء نفس وظيفة هذه الوسيلة، باستخدام الوسيلة CreateCommand الخاصة بكائن الاتصال، وفي هذه الحالة ستختصر السطر الثاني من الكود السابق:

```
Dim Command = Cn.CreateCommand
```

- **إنشاء معامل CreateParameter:**

تعيد معاملا من النوع العام DbParameter، لكنه يكون مخصصا للتعامل مع مزود البيانات الذي أنشأت المصنع الحالي للتعامل معه.
ويمكنك أداء نفس الوظيفة، باستخدام الوسيلة CreateParameter الخاصة بكائن الأمر.

- **إنشاء موصل بيانات CreateDataAdapter:**

تعيد موصل بيانات من النوع العام DbDataAdapter، لكنه يكون مخصصا للتعامل مع مزود البيانات الذي أنشأت المصنع الحالي للتعامل معه.. وقد استخدمنا هذه الوسيلة في الدالة GetTable في المشروع Factories كالتالي:

```
Dim Table As New DataTable
Dim Da = Fac.CreateDataAdapter
Da.SelectCommand = Cmd
Da.Fill(Table)
```

- **إنشاء باني أوامر CreateCommandBuilder:**

تعيد باني أوامر من النوع العام DbCommandBuilder، لكنه يكون مخصصا للتعامل مع مزود البيانات الذي أنشأت المصنع الحالي للتعامل معه.

- **إنشاء عداد مصادر البيانات CreateDataSourceEnumerator:**

تعيد عداد مصادر البيانات من النوع العام DbDataSourceEnumerator، لكنه يكون مخصصا للتعامل مع مزود

البيانات الذي أنشأت المصنع الحالي للتعامل معه.. لاحظ أن مزود سيكويل سيرفر هو الوحيد الذي يدعم هذه الإمكانية، لأن قواعد البيانات الخاصة به تعمل على خادم، لهذا ستعيد هذه الوسيلة Nothing إذا استخدمتها مع أي مزود بيانات آخر غير سيكويل سيرفر!

ويمكنك أن تستخدم الخاصية CanCreateDataSourceEnumerator أولا قبل استدعاء هذه الوسيلة، لتعرف إن كان المزود يدعم عداد المصادر أم لا.

- **إنشاء تصريح CreatePermission:**

تعيد تصريحا من النوع العام CodeAccessPermission، لكنه يكون مخصصا للتعامل مع مزود البيانات الذي أنشأت المصنع الحالي للتعامل معه... لاحظ أن الفئة DBDataPermission ترث الفئة CodeAccessPermission، ومنه تشتق فئات التصريح الخاصة بكل مزود بيانات مثل SqlClientPermission.. شرح هذه المواضيع خارج نطاق هذا الكتاب.

وترث الفئات التالية الفئة DbProviderFactory:

١- OdbcFactory Class.
٢- OleDbFactory Class.
٣- OracleClientFactory Class.
٤- SqlClientFactory Class.

ولا يوجد جديد في هذه الفئات يستحق شرحه، فهي تملك نفس وسائل الفئة الأم، لكن مع فارق واحد: أنها تعيد أنواعا خاصة بكل مزود، بدلا من الأنواع العامة التي تعيد وسائل الفئة الأم.

وخير طريقة لإدراك عبقرية مصانع المزودات، هي أن نعيد كتابة المشروع DbTasks بطريقة عامة، تسمح بالتعامل مع أي مزود بيانات.. كما تذكر، فقد أنشأنا في هذا المشروع فئة اسمها MyDbConnector تسهل علينا إجراء أي عملية على قواعد بيانات سيكويل سيرفر.. الآن حان الوقت لنعمم هذه الفئة، بحيث نستطيع استخدامها للتعامل مع باقي أنواع المزودات التي يدعمها إطار العمل.. هذا هو ما فعلناه في المشروع Factories، الذي هو نسخة طبق الأصل من المشروع DbTasks، لكنه يعرض على النموذج زري تحويل Radio Buttons ليستطيع المستخدم اختيار التعامل مع قاعدة بيانات آكسيس أو

قاعدة بيانات سيكويل سيرفر، والرائع حقا أن كـود الأزرار الموضـوعة علـى النموذج ظل كما هو بدون تغيير، بفضل استخدام مصانع المزودات!
لكننـا بـالطبع أجرينـا بعـض التغييـرات الـضرورية علـى كـود الفئـة MyDbConnector، فقد عرفنا فيها مرقما اسمه Providers، واستخدمناه في تعريف معامل ثان لحدث الإنشاء New، ليرسل المستخدم عند إنشاء نسخة مـن هذه الفئة، نص الاتصال ونوع المزود الذي يريد التعامل معه.
كما عرفنا دالة اسمها GetProviderName، تستقبل قيمة المرقم Providers، وتعيـد الـنص الـذي يمثـل اسـم هـذا المـزود، لنرسـله إلـى الوسـيلة DbProviderFactories.GetFactor للحصول على مصنع المزود الذي يريد المستخدم التعامل معه.. بعد هذا يصير من السهل استخدام وسائل هذا المـصنع للحصول على كائن الاتصال وكائن الأمر وموصل البيانات اللازمة للتعامل مـع قاعدة البيانات.
ولو نظرت إلى كود الفئة MyDbConnector في هذا المـشروع، فـستجد أن التعديلات التي أدخلناها طفيفة، لكن تأثيرها هائل، فقد صارت لدينا فئـة عامـة تستطيع أداء معظم ــ إن لم يكن كل ــ الوظائف التي نريدها على أي نوع مـن أنواع قواعد البيانات، مما يتيح لك استخدامها في مشاريعك لتقليل الكـود الـذي تكتبه إلى أقل حد ممكن!.
لاحظ أننا نغير نوع قاعدة البيانات التي نتعامل معها، في حدث تغيـر الاختيـار CheckedChanged الخاص بزري التحويل، وذلك بالكود البسيط التالي:

```
If RdSql.Checked Then
   DbBooks = New MyDbConnector(
             My.Settings.BooksMdfConStr,
             MyDbConnector.Providers.SqlServer)
Else
  DbBooks = New MyDbConnector(
             My.Settings.BooksMdbConStr,
             MyDbConnector.Providers.OleDb)
End If
```

حيث DbBooks هو متغير معرف على مستوى النموذج، نضع فيه نسخة الفئة MyDbConnector التي نستخدمها لتنفيذ وظائف الأزرار.
لاحظ أيضا أن استخدام الزر "الكتب١" لاستدعاء الإجراء المخزن، يستلزم منـك أولا أن تستخدم المشروع AccessStoredProcedure لإضافة الإجراء المخزن GetAuthorBooks إلى قاعدة بيانات الكتب الخاصة بآكسيس.

**الطبقات المتعددة N-Tiers:**

لعل المشروع السابق يكشف لك أهمية تقسيم مشاريع قواعد البيانات إلى طبقات Layers مستقلة عن بعضها.. هذا يسهل عليك تطوير أي طبقة دون تغيير أي شيء في الطبقات الأخرى.. فنحن هنا مثلا عدلنا كود الفئة MyDbConnector دون أن نغير أي شيء تقريبا في الكود الذي يستخدمها، وهي ميزة تتضح فوائدها الهائلة في المشاريع الضخمة، التي تريد الاستفادة من التطويرات التي تحدث في تقنيات قواعد البيانات، دون إعادة كتابة الكود كله منذ البداية.. فلو أن هذه المشاريع مقسمة إلى طبقات، فسينحصر التطوير على طبقة الاتصال بقاعدة البيانات للاستفادة من التقنيات الجديدة، بينما ستظل الطبقة التي تعرض البيانات للمستخدم كما هي بدون تغيير يذكر.. وتسمى البرامج التي تستخدم هذا التنظيم باسم التطبيقات متعددة الطبقات Multi-Tier Applications أو n-Tier Applications، وتسمى أيضا باسم التطبيقات الموزعة Distributed Applications لأنها مقسمة على أكثر من طبقة.. والشهير أن تكتب مشاريع قواعد على ثلاث طبقات:

١- طبقة البيانات Data Tier:

وهي الطبقة التي توجد فيها قاعدة البيانات بما فيها من جداول وعلاقات وإجراءات مخزنة، ومستخدمين وصلاحيات وقواعد سرية وحماية وصيانة وحفظ نسخ احتياطية من البيانات.. إلخ.. وفي الشركات الكبيرة يكون هناك موظفون مسئولون عن إدارة هذه الطبقة.

٢- طبقة التعامل مع البيانات Data Access Tier:

في هذه الطبقة، يوجد الكود الذي يتصل بقاعدة البيانات ويحضر النتائج منها.. والفئة MyDbConnector هي مجرد مثال مبسط على هذه الطبقة، لكن دوت نت تمنحك إمكانيات أقوى لتصميم هذه الطبقة مثل مجموعة البيانات محددة النوع وموصلات الجداول التي سنتعرف عليها في الفصل القادم، ومثل LinQ-To-SQL وغير ذلك.

وتمتاز هذه الطبقة بأنك تستطيع استخدامها في أكثر من مشروع، مما يوفر لك الوقت والجهد، كما يمكنك تطويرها دون الحاجة إلى إعادة كتابة المشاريع التي تعتمد عليها.

٣- طبقة عرض البيانات Data Display Tier:

في هذه الطبقة يوجد الكود الذي يعرض البيانات للمستخدم، والمفروض ألا يوجد في هذه الطبقة أي كود يتصل بقاعدة البيانات.

وسنتعرف على التطبيقات متعددة الطبقات بإذن الله بصورة أشمل، في الكتاب المخصص للمواضيع المتقدمة في برمجة قواعد البيانات.

## فئة عداد مصادر البيانات
## DbDataSourceEnumerator Class

هذه الفئة أساسية مجردة، لكن حتى الآن لا ترثها إلا الفئة SqlDataSourceEnumerator، لأن قواعد بيانات سيكويل سيرفر هي التي تعمل على خادم، سواء أكان خادما محليا Local أو بعيدا Remote.
وتتيح لك هذه الفئة الحصول على معلومات عن الخوادم المتوفرة حاليا على الشبكة التي يتصل بها جهاز العميل.
وتمتلك هذه الفئة الوسيلة الوحيدة التالية:

### معرفة مصادر البيانات GetDataSources:

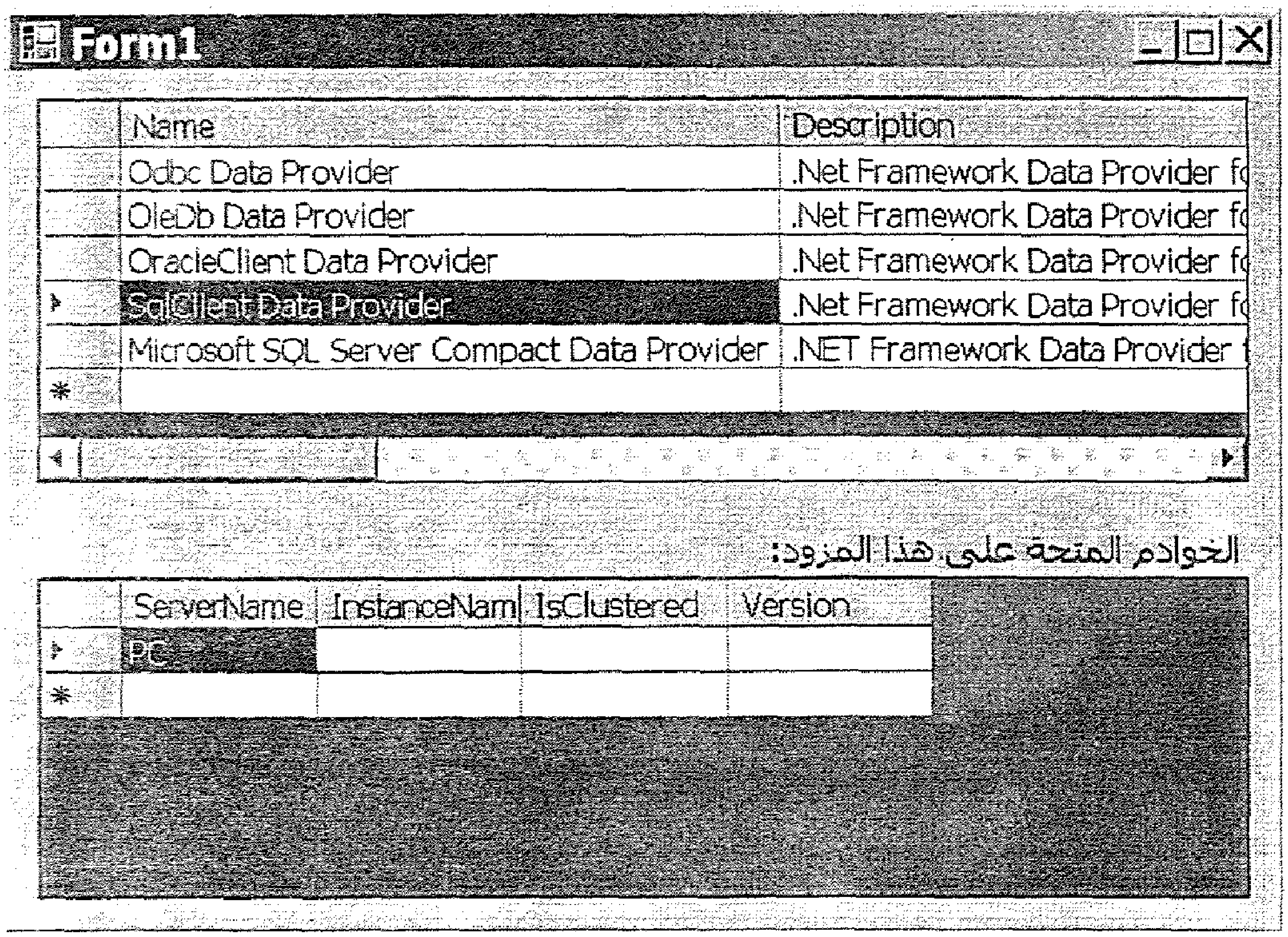
تعيد جدول بيانات DataTable، يحتوي على صفوف فيها تفاصيل الخوادم المتاحة.. ويعرض هذا الجدول الأعمدة التالية:

| | |
|---|---|
| ServerName | اسم خادم البيانات. |
| InstanceName | اسم النسخة التي تعمل من الخادم.. لاحظ أن سيكويل سيرفر يتيح تشغيل أكثر من نسخة من الخادم. |
| IsClustered | True إذا كان الخادم جزءا من تجمع Cluster من الخوادم. |
| Version | إصدار الخادم. |

ويمكنك استخدام هذه الوسيلة لتعرض للمستخدم قائمة بأسماء الخوادم المتاحة، ليختار الخادم الذي يريد أن يتصل به.. لكن عليك أن تلاحظ ما يلي:

١- هذه الوسيلة تستهلك وقتا عند تنفيذها، بسبب بحثها عن الخوادم المتاحة على الشبكة.
٢- ناتج هذه الوسيلة قد يختلف من مرة إلى أخرى، بسبب ظهور بعض الخوادم أو اختفائها!
٣- هذه الوسيلة قد لا تعيد كل الخوادم المتاحة فعلا، لهذا عليك أن تعرض للمستخدم مربع نص أيضا، ليكتب اسم الخادم بنفسه إذا لم يجده في القائمة.

وقد استخدمنا هذه الوسيلة في المشروع DataProviders لنعرض في الجدول السفلي، الخوادم المتاحة على المزود المحدد في الجدول العلوي، كما هو موضح في الصورة:




لفعل هذا، استخدمنا الحدث RowEnter الخاص بجدول العرض، وفيه استخدمنا رقم الصف للحصول على كائن صف البيانات DataRow المناظر له في جدول المزودات، وأرسلنا هذا الصف إلى الوسيلة DbProviderFactories.GetFactory للحصول على مصنع مزود البيانات:

**Dim R = TblProviders.Rows(e.RowIndex)**
**Dim Pf = DbProviderFactories.GetFactory(R)**

بعد هذا استخدمنا الوسيلة CanCreateDataSourceEnumerator للتأكد من أن المزود يتيح عرض الخوادم، ومن استخدمنا الوسيلة CreateDataSourceEnumerator للحصول على عداد الخوادم، ومنه حصلنا على الجدول الذي يحتوي على تفاصيل هذه الخوادم باستخدام الوسيلة GetDataSources وعرضناه في جدول العرض:

```
If Pf.CanCreateDataSourceEnumerator Then
   Dim Se = Pf.CreateDataSourceEnumerator
   Dim TblServers = Se.GetDataSources
   DgServers.DataSource = TblServers
Else
   DgServers.DataSource = Nothing
End If
```

عند تجربة هذا البرنامج على جهازك، لن تظهر أية خوادم إلا عند اختيار مزود سيكويل سيرفر، حيث سيظهر الخادم المحلي Local Server المعرف على جهازك (وهو يمتلك نفس اسم جهازك) وفي الغالب لن يظهر الخادم SQLEXPRESS الذي يعمل على هذا الخادم المحلي!

## فئة عداد مصادر بيانات سيكويل سيرفر
## SqlDataSourceEnumerator Class

هذه الفئة موجودة في النطاق System.Data.Sql، وهي ترث الفئة DbDataSourceEnumerator.

وتتعامل هذه الفئة مع عداد مخصص للمرور عبر خوادم سيكويل سيرفر المتوفرة على الشبكة الحالية.

وتملك هذه الفئة خاصية واحدة جديدة، وهي:

### النسخة Instance:

تعيد نسخة جديدة من الفئة SqlDataSourceEnumerator، مما يغنيك عن استخدام مصنع المزود أولا للوصول إليها.

والمشروع SqlServers يريك كيف يمكن استخدام هذه الخاصية لعرض الخوادم المتوفرة على جهازك، وهو لا يحتاج لفعل هذا، إلا إلى هذا السطر الوحيد من الكود:

```
DgServers.DataSource =
   SqlDataSourceEnumerator.Instance.GetDataSources
```

-١٢-

# مجموعة البيانات DataSet

مجموعة البيانات هي وعاء مصغر لقاعدة البيانات في برنامجك، يتيح لك أن تحمل في الذاكرة، بعض الجداول أو أجزاء منها (تبعا للاستعلام المستخدم) مع قدرتك إنشاء العلاقات بينها ووضع القيود عليها، وبهذا يمكنك التعامل مع هذه البيانات على جهازك بعد قطع الاتصال مع الخادم.

وتمتاز مجموعة البيانات بأنها عامة، فهي تستطيع التعامل مع أي نوع من أنواع قواعد البيانات، ويمكنها استيعاب الجداول دون أن يعنيها مصدرها، بينما تترك مهمة التعامل مع مصدر البيانات لموصل البيانات وكائن الاتصال.. لهذا يوجد في دوت نت نوع واحد فقط من مجموعة البيانات، على عكس الكائنات التي تعرفنا عليها سابقا، والتي يوجد منه نوع خاص بكل مزود.

وتستخدم مجموعة البيانات داخليا كود XML لحفظ مخططات الجداول والأعمدة Schema، وبيانات الصفوف.. ويتيح لك هذا حفظ مخططات وبيانات الجداول من مجموعة البيانات إلى جهازك في صورة وثائق XML، ومن ثم إعادة تحميلها في مجموعة البيانات مرة أخرى لاحقا.

وكما ذكرنا من قبل، تحتفظ مجموعة البيانات بنسختين من كل سجل:

١- النسخة الأصلية Original Version التي تم تحميلها من قاعدة البيانات.

٢- النسخة الحالية Current Version التي تحتوي على السجل بعد حدوث تغييرات به.

ويمتلك كل سجل الخاصية RowState التي توضح حالته، وهل دخلت عليه تغييرات وهل تم حفظ هذه التغييرات إلى قاعدة البيانات أم لا.. وبهذا التنظيم تستطيع مجموعة البيانات نقل التغيرات إلى قاعدة البيانات وتحديثها، بالاعتماد على موصل البيانات، الذي يعيد فتح الاتصال مع الخادم.. هذا يجعل مجموعة البيانات أفضل من قارئ البيانات في الحالات التالية:

١- إذا كان البرنامج يتعامل مع الكثير من الجداول والسجلات، ويستخدمها أكثر من مرة بدون ترتيب معين.. في هذه الحالة يكون المرور عبرها على التوالي باستخدام قارئ البيانات أمرا غير عملي.
٢- إذا كان المطلوب عرض البيانات للمستخدم والسماح له بالتعامل معها وتعديلها بحرية، والإضافة إليها والحذف منها.. أنت تعرف أن قارئ البيانات لا يقوم بتحديث السجلات، فهو للقراءة فقط.
٣- إذا كانت هناك علاقات بين الجداول وقيود مفروضة عليها، ومن المهم التعامل معها في البرنامج عند الإضافة والحذف، فمجموعة البيانات تسمح بالتعامل مع العلاقات والقيود، وهذا غير متوفر في قارئ البيانات.

لكن على الجانب الآخر، تعاني مجموعة البيانات من العيبين التاليين:
١- تعتبر مجموعة البيانات عبئا على ذاكرة الجهاز، لهذا يجب عليك تحميلها بأقل قدر ممكن تحتاجه من البيانات، ولا تضع فيها الجداول بكامل صفوفها بدون فائدة، وبدلا من هذا استخدم شرطا في جملة التحديد SELECT لتحصل على السجلات المطلوبة بالضبط. أيضا، لا تحمل من الجداول أعمدة لا يحتاجها المستخدم.
٢- قد تسبب مجموعة البيانات مشاكل عند تحديث قاعدة البيانات، وذلك إذا كان مستخدمون آخرون قد غيروا قيم بعض السجلات في قاعدة البيانات أثناء قطع الاتصال وتعاملك معها في مجموعة البيانات، فيما يسمى بمشاكل التطابق Concurrency Violations.. وقد رأينا في الفصل السابق كيف يمكن حل هذه المشكلة.

والآن، دعنا نتعرف على فئة مجموعة البيانات.

## فئة مجموعة البيانات DataSet Class

هذه الفئة توجد في النطاق System.Data وهي تمثل واجهة مصدر القائمة IListSource التي سنتعرف عليها لاحقا.
ولحدث إنشاء هذه الفئة الصيغتان التاليتان:
١- الصيغة الأولى بدون معاملات.
٢- الصيغة الثانية تستقبل معاملا نصيا، يمثل اسم مجموعة البيانات، الذي سيستخدم عند حفظ مجموعة البيانات في ملف XML.

وتمتلك هذه الفئة الخصائص التالية:

**اسم مجموعة البيانات DataSetName:**
تحدد اسم مجموعة البيانات، ليتم استخدامه كاسم لعنصر الوثيقة Document Element في كود XML عند حفظ مجموعة البيانات.

**نطاق الاسم Namespace:**
تحدد اسم النطاق الذي سيتم تحته حفظ مجموعة البيانات في كود XML.

**البادئة Prefix:**
تحدد البادئة التي ستستخدم لتمييز العناصر التي تنتمي إلى نطاق مجموعة البيانات.. هذا مفيد إذا كان ملف XML يحتوي على نطاق الاسم فيه أكثر من مجموعة بيانات، وتريد تمييز كل منها تحت نطاق فرعي خاص بها.

| **ملحوظة:** |
|---|
| عند استخدام الوسيلتين ReadXml و ReadXmlSchema لتحميل البيانات أو المخطط في مجموعة البيانات، فإنهما تبحثان في ملف XML عن نطاق الاسم الموضح في الخاصية DataSetName، ومجوعة البيانات المميزة بالبادئة الموضحة في الخاصية Prefix، فإذا لم تعثر في الملف عن مجموعة بيانات تحقق هذين الشرطين، لا يتم تحميل أي شيء من الملف. |

ويريك المشروع DataSetSample مثالا على استخدام هذه الخصائص.. ستجد هذا الكود مثلا حدث تحميل النموذج:

```
Ds.Namespace = "My Project"
Ds.Prefix = "Authors-Books"
Ds.DataSetName = "DsBooks"
```

ويظهر تأثير هذه الخصائص عند ضغط الزر "حفظ المخطط في ملف"، حيث ستجد أسماء هذه الخصائص مستخدمة في تعريف مخطط مجموعة في الملف C:\DsBooksSchema.xml.

**حساسة لحالة الأحرف CaseSensitive:**

إذا جعلت قيمة هذه الخاصية True، فستصير عمليات المقارنة والترشيح Filtering حساسة لحالة الأحرف.. هذا يؤثر في نتائج الوسيلة DataTable.Select والخاصية DataColumn.Expression.. والقيمة الافتراضية لهذه الخاصية هي False.

لاحظ أن تغيير قيمة هذه الخاصية، سيغير تلقائيا قيمة الخاصية CaseSensitive الخاصة بكل جدول في مجموعة البيانات.

**المحل Locale:**

تقرأ أو تغير كائن معلومات الثقافة CultureInfo، الذي يحتوي على تفاصيل اللغة التي تستخدم لمقارنة وترتيب النصوص الموجودة في جداول مجموعة البيانات.

لاحظ أن تغيير قيمة هذه الخاصية، سيغير تلقائيا قيمة الخاصية Local الخاصة بكل جدول في مجموعة البيانات.

**فرض القيود EnforceConstraints:**

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتم التأكد من صحة القيود المفروضة على الجداول عند إجراء عمليات التحديث والإدراج والحذف.

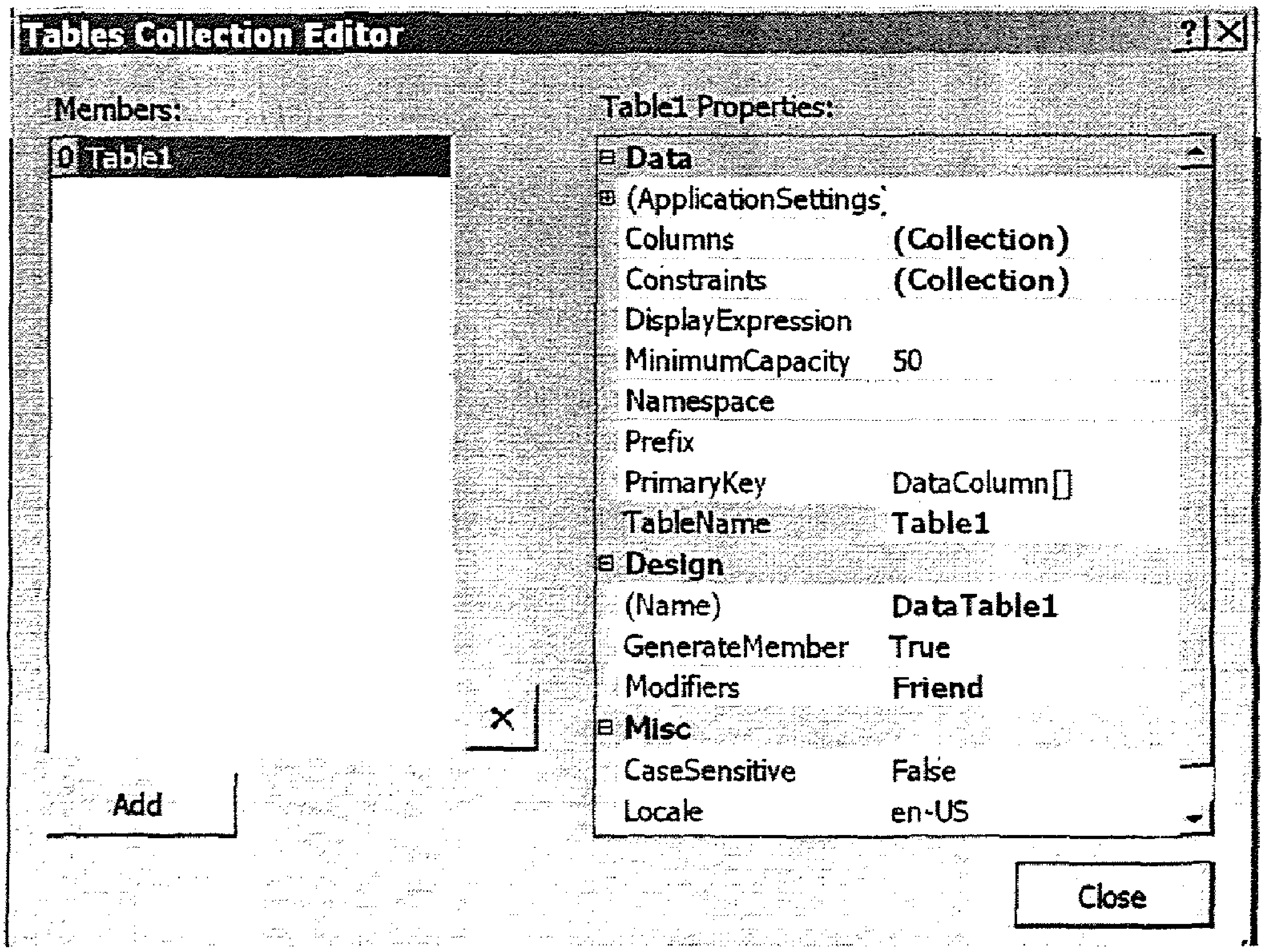
**توجد بها أخطاء HasErrors:**

تعيد True إذا كان أي من جداول مجموعة البيانات قد حدثت به أخطاء أثناء عملية التحديث.. ويمكنك فحص الخاصية HasErrors الخاصة بكل جدول لمعرفة الجدول الذي تسبب في الخطأ.

### الجداول Tables:

تعيد مجموعة جداول البيانات DataTableCollection، التي تحتوي على كائنات الجداول DataTable Objects الموجودة في مجموعة البيانـــات.. وسنتعرف على هذه المجموعة بالتفصيل في الفصل التالي.

ما يعنيا هنا هو أنك تستطيع تحرير هـــذه المجموعـــة مـــن خـــلال نافـــذة الخصائص.. فلو ضغطت زر الانتقال الموجود في خانة قيمة هذه الخاصية، فستظهر لك نافذة محرر مجموعة الجـــداول Table Collection Editor كما هو موضح في الصورة:

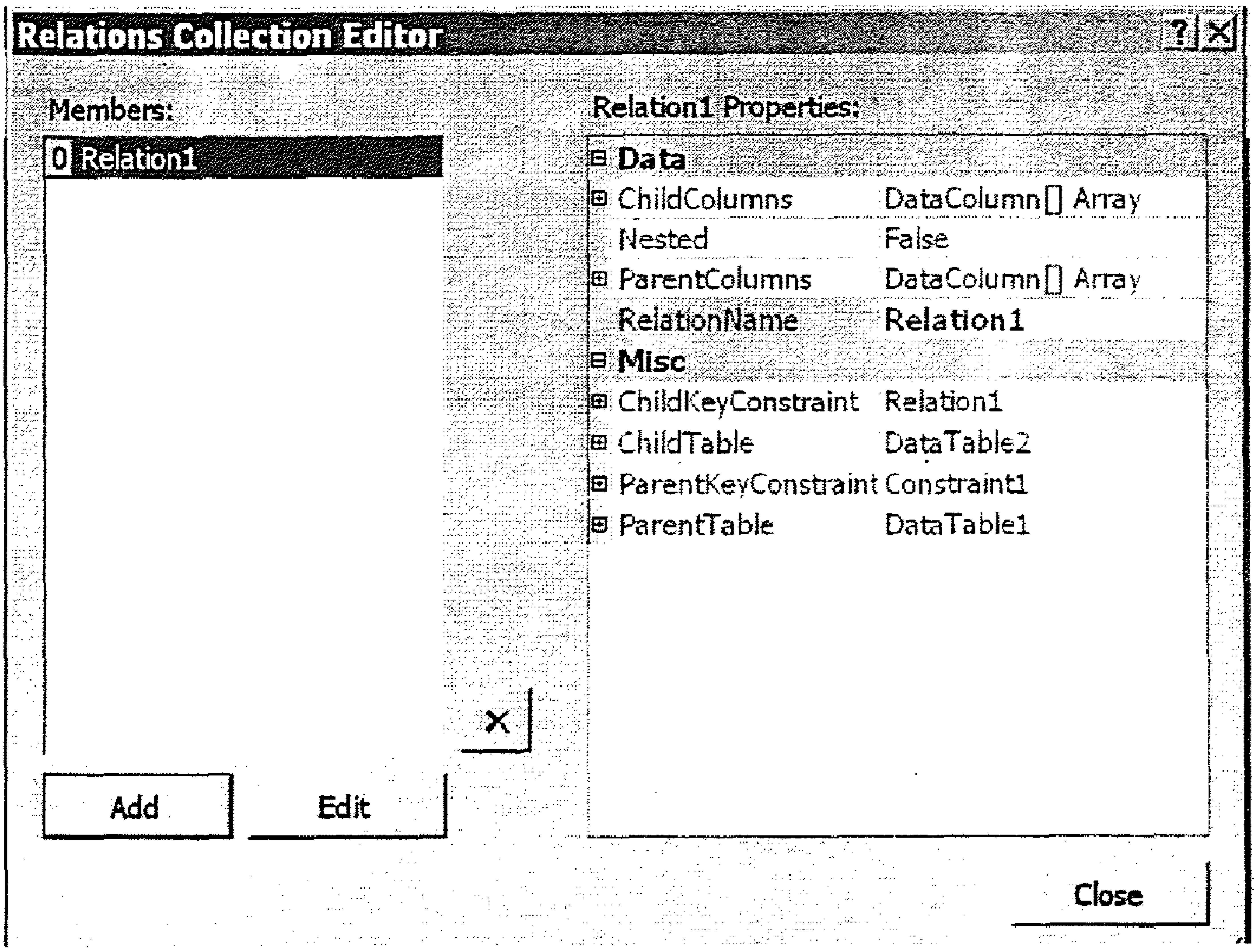



في هذه النافذة يمكنك إضافة جداول جديد بضغط الزر Add، ثم اســـتخدام الخصائص الموجودة في القسم الأيمن من النافذة لتغيير اسم هـــذا الجـــدول وطريقة عرضه والأعمدة الموجودة به.. وسنتعرف على هذه الخـــصائص بالتفصيل في الفصل اللاحق.

## العلاقات Relations:

تعيد مجموعة علاقات البيانات DataRelationCollection، التي تحتوي على كائنات العلاقات DataRelation Objects الموجودة في مجموعة البيانات.. وسنتعرف على هذه المجموعة بالتفصيل في الفصل التالي.

لاحظ أن ملء مجموعة البيانات بالجداول والسجلات لا يضيف العلاقات بين الجداول تلقائيا إلى مجموعة العلاقات Relations.. لهذا عليك أن تضيف هذه العلاقات بنفسك إلى مجموعة العلاقات، سواء من الكود أو باستخدام نافذة المخطط كما سنرى لاحقا.. كما يمكنك إضافة العلاقات بطريقة مرئية من خلال نافذة الخصائص.. فلو ضغطت زر الانتقال الموجود في خانة قيمة هذه الخاصية، فستظهر لك نافذة محرر مجموعة العلاقات Relations Collection Editor كما هو موضح في الصورة:




اضغط الزر Add لإضافة علاقة جديدة.. ستظهر نافذة إنشاء العلاقة لتسمح لك بتحديد الجدولين والأعمدة المشتركة في العلاقة كما تعلمنا من قبل.. وبعد أن تضغط OK لإغلاق نافذة العلاقة، ستظهر العلاقة في القائمة اليسرى، وتظهر خصائصها في القائمة اليمنى.. ولو أردت تغيير عناصر العلاقة ، فاضغط الزر Edit لعرض نافذة العلاقة مرة أخرى.

### مدير العرض الافتراضي DefaultViewManager:

تعيد كائن مدير عرض البيانات DataViewManager Object الذي يتحكم في البيانات التي تعرضها مجموعة البيانات.. وسنتعرف على هذا الكائن بالتفصيل لاحقا.

### الخصائص الإضافية ExtendedProperties:

تعيد مجموعة الخصائص PropertyCollection التي تحتوي على الخصائص الإضافية التي تضيفها إلى مجموعة البيانات.
والمجموعة PropertyCollection ترث الجدول المختلط Hashtable، مما يتيح لك إضافة اسم الخاصية كمفتاح Key، والقيمة التي تريد حفظها فيها كقيمة Value.
والمثال التالي يضيف إلى مجموعة البيانات خاصية إضافية تحتفظ باسم البرنامج الخاص بك، ثم يعرض قيمة هذه الخاصية في رسالة:

```
Ds.ExtendedProperties.Add("ProgName", "MyProg")
MsgBox(Ds.ExtendedProperties("ProgName"))
```

### تنسيق التراسل RemotingFormat:

تحدد التنسيق الذي سيتم به إرسال البيانات من جهاز إلى آخر، عندما تتعامل مجموعة البيانات مع برنامج يستخدم التحكم عن بعد Remoting، وهي تأخذ إحدى قيمتي المرقم SerializationFormat التاليتين:

| | |
|---|---|
| Xml | يتم إرسال البيانات في صورة نصوص XML.. هذه هي القيمة الافتراضية. |
| Binary | يتم إرسال البيانات في صورة أرقام ثنائية Binary.. هذا متاح فقط بدءا من إصدار إطار العمل الثاني. |

لاحظ أن تغيير قيمة هذه الخاصية، يغير قيمة الخاصية RemotingFormat الخاصة بكل جدول في مجموعة البيانات.

### طريقة سلسلة المخطط SchemaSerializationMode:

تحدد كيف سيتم التعامل مع مخطط البيانات عند سلسلة مجموعة البيانات محددة النوع Typed DataSet.. والسلسلة Serialization: هي تحويل محتويات كائن موجود في الذاكرة إلى بيانات يمكن حفظها في ملف أو إرسالها عبر الشبكة.. هذا يتيح لك الاحتفاظ بحالة الكائن بعد إغلاق البرنامج

لاستعادتها مرة أخرى بعد إعادة تشغيله، أو إرسالها إلى جهاز آخر للتحكم في هذا الكائن عن بعد Remoting.. وسنتعرف على السلسلة Serialization والتحكم البعيد Remoting بالتفصيل في كتاب خاص بالمواضيع المتقدمة في برمجة إطار العمل بإذن الله.
وتأخذ هذه الخاصية إحدى قيمتي المرقم SchemaSerializationMode:

| | |
|---|---|
| IncludeSchema | إضافة مخطط البيانات ضمن عملية السلسلة.. هذه هي القيمة الافتراضية. |
| ExcludeSchema | عدم إضافة مخطط البيانات ضمن عملية السلسلة، مع الاكتفاء بسلسلة خصائص مجموعة البيانات التي تغيرت قيمها عن القيمة الافتراضية، وبهذا يتم تقليل حجم البيانات المسلسلة بشكل كبير.. لاحظ أن هذه القيمة لا تصلح إذا كانت مجموعة البيانات عادية (غير محددة النوع Un-typed DataSet). |

وسنتعرف على مجموعة البيانات محددة النوع لاحقا في هذا الفصل.

كما تمتلك هذه الفئة الوسائل التالية:

**محو Clear:**
تمحو كل السجلات من كل جداول مجموعة البيانات، لكنها لا تمحو الجداول نفسها، ولا العلاقات بينها.. لاحظ أن هذه الوسيلة ستسبب خطأ في البرنامج لو كانت مجموعة البيانات تتعامل مع وثيقة XML من النوع XmlDataDocument.

**تصفير Reset:**
تفرغ مجموعة البيانات من جميع محتوياتها، بما في ذلك الجداول والعلاقات والقيود.

**نسخ Clone:**
تنسخ تركيب مجموعة البيانات (مخططات الجداول، والعلاقات والقيود) إلى مجموعة بيانات جديدة وتعيد مرجعا إليها.. لكنها لا تنسخ أي سجلات.

### نسخ Copy:

تنسخ مجموعة البيانات كاملة (مخططات الجداول، والعلاقات والقيود والسجلات أيضا) إلى مجموعة بيانات جديدة وتعيد مرجعا إليها.

### معرفة التغيرات GetChanges:

تعيد مجموعة بيانات جديدة، تحتوي جداولها على الصفوف التي تمّ تعديلها أو إضافتها أو حذفها منذ ملء مجموعة البيانات الأصلية، أو منذ آخر استدعاء للوسيلة AcceptChanges.. وتعيد هذه الوسيلة Nothing إذا لم تجد أية تغيرات في مجموعة البيانات.

لاحظ أن هذه الوسيلة قد تضيف إلى مجموعة البيانات الجديدة بعض الصفوف التي لم تتغير بياناتها، وذلك للمحافظة على صحة العلاقات والقيود بين الجداول، مما يتيح لك إعادة دمج مجموعة البيانات الجديدة بمجموعة البيانات الأصلية إذا أردت، دون حدوث أية أخطاء.

وتوجد صيغة ثانية لهذه الوسيلة، تستقبل إحدى قيم المرقم DataRowState، التي تمكنك من الحصول على السجلات التي حدث بها نوع محدد من التغيير دون سواه.. وهذه القيم هي:

| | |
|---|---|
| مستقلّ Detached | تمّ إنشاء هذا السجلّ ولكنه لم يوضع بعد في مجموعة السجلات Rows الخاصة بأي جدول، أو أنه حذف للتو من مجموعة سجلات أحد الجداول. |
| لم يتغير Unchanged | لم تتغير بيانات هذا السجلّ، منذ أن تمّ تحميله من قاعدة البيانات أو منذ آخر استدعاء للوسيلة AcceptChanges. |
| مُضاف Added | هذا السجلّ ليس موجودا في قاعدة البيانات، وإنما تمّت إضافته كسجل جديد إلى مجموعة البيانات. |
| محذوف Deleted | تمّ حذف هذا السجلّ من مجموعة البيانات، ولكنه ما زال موجودا في قاعدة البيانات. |
| معدّل Modified | تمّ تعديل هذا السجلّ، ولكن لم يتمّ حفظ التعديلات في قاعدة البيانات بعد. |

ويمكنك دمج أكثر من قيمة من قيم هذا المرقم معا، باستخدام المعامل OR.

### تم تغييرها HasChanges:

تعيد True، إذا كانت مجموعة البيانات تحتوي على سجلات قد تم تعديلها أو إضافتها أو حذفها، ولم تحفظ بعد في قاعدة البيانات.
ويمكنك استخدام هذه الوسيلة في حدث إغلاق النموذج FormClosing، لسؤال المستخدم إن كان يريد حفظ البيانات قبل إغلاق البرنامج أم لا، وهو ما فعلناه في المشروع CustomDataSet.. وقد استخدمنا هذه الوسيلة في الزر "تحميل من قاعدة البيانات" في المشروع DataSetSample، لحفظ أية تغييرات قبل إعادة تحميل البيانات.
وتوجد صيغة ثانية لهذه الوسيلة، تستقبل إحدى قيم المرقم DataRowState التي تعرفنا عليها من قبل.. وتعيد هذه الصيغة True إذا كانت مجموعة البيانات تحتوي على سجلات وقع عليها نوع التغيير المرسل كمعامل.. والجملة التالية تخبرك إن كانت هناك سجلات جديدة أضيفت إلى مجموعة البيانات أم لا:

**MsgBox(Ds.HasChanges(DataRowState،Added))**

### قبول التغييرات AcceptChanges:

تجبر كل جداول مجموعة البيانات على استدعاء الوسيلة AcceptChanges الخاصة بها.

### رفض التغييرات RejectChanges:

تجبر كل جداول مجموعة البيانات على استدعاء الوسيلة RejectChanges الخاصة بها.

### إنشاء قارئ بيانات CreateDataReader:

تعيد قارئ بيانات الجداول DataTableReader، الذي يمكنك من خلاله المرور عبر سجلات كل جداول مجموعة البيانات.
وتنشئ هذه الوسيلة مجموعة نتائج Result Set لكل جدول، بنفس ترتيب الجداول في مجموعة الجداول DataSet.Tables، وإذا كان أحد الجداول خاليا من السجلات، فستوضع مقابله في قارئ البيانات مجموعة نتائج فارغة، وذلك للحفاظ على الترتيب.. ويمكنك الانتقال من قراءة سجلات جدول إلى سجلات الجدول التالي باستخدام الوسيلة NextResult الخاصة بقارئ البيانات كما تعلمنا من قبل.

وتوجد صيغة ثانية لهذه الوسيلة تتيح لك التحكم في ترتيب النتائج، حيث تستقبل مصفوفة جداول DataTable Array، تحتوي على جداول مجموعة البيانات التي تريد أن تقرأها، مع ملاحظة أن الجدول الذي يظهر في هذه المصفوفة أولا سيعرض قارئ البيانات سجلاته أولا.
ويريك الزر "إنشاء قارئ بيانات" كيف يمكنك استخدام هذه الوسيلة لعرض كل محتويات مجموعة البيانات في نافذة المخرجات Output Window.

### تحميل Load:

تتيح لك هذه الوسيلة استخدام قارئ البيانات DataReader لإضافة المزيد من السجلات إلى مجموعة البيانات.. ولهذه الوسيلة ثلاث صيغ:

١- الصيغة الأولى تستقبل ثلاثة معاملات:

- معامل من نوع الواجهة IDataReader يستقبل قارئ البيانات.
- إحدى قيم المرقم LoadOption التي تحدد ماذا سيحدث إذا كانت بعض السجلات موجودة سابقا في مجموعة البيانات، وهل سيتم تحديث النسخة الأصلية من السجل Original Version أم النسخة الحالية Current Version.. وقد تعرفنا على هذا المرقم في الفصل السابق.
- مصفوفة جداول DataTable Array، تحتوي على بعض الجداول الموجودة في مجموعة الجداول DataSet.Tables، ليتم ملؤها بالسجلات من قارئ البيانات، حيث ستوضع سجلات كل مجموعة من النتائج ResultSet في الجدول المناظر لها في الترتيب في المصفوفة.

٢- الصيغة الثانية مماثلة للصيغة السابقة، إلا أن معاملها الثالث يستقبل مصفوفة نصية تحتوي على أسماء الجداول بدلا من كائنات الجداول.

٣- الصيغة الثالثة تزيد بمعامل إضافي على الصيغة الأولى.. هذا المعامل يأتي في الموضع الثالث في ترتيب المعاملات، وهو مندوب Delegate من النوع FillErrorEventHandler، وهو المندوب المستخدم في تعريف الحدث FillError الخاص بموصل البيانات.. ويمكنك أن ترسل إلى هذا المندوب عنوان إجراء مناسب، ليتم استدعاؤه لو حدث خطأ عند إضافة أحد السجلات إلى مجموعة البيانات.

## دمج Merge:

تمزج بعض السجلات بسجلات مجموعة البيانات.. والمزج يعني أنّ السجلات الجديدة ستتمّ إضافتها إلى مجموعة البيانات، أمّا السجلات الموجودة سابقا، فسيتمّ وضع السجلات المضافة بدلا منها.. وتتم مطابقة السجلات من خلال المفتاح الأساسي لكل منها، لهذا يجب أن يحتوي جدول مجموعة البيانات على مفتاح أساسي، وإلا أدت عملية الدمج إلى تكرار نفس الصفوف مرتين.

ولهذه الوسيلة العديد من الصيغ:

١- بعض الصيغ ذات معامل واحد، يستقبل البيانات المراد مزجها، سواء كانت قادمة من مجموعة بيانات DataSet، أو جدول DataTable أو مصفوفة سجلات DataRow Array.

٢- بعض الصيغ تزيد على الصيغ السابقة بمعامل ثان، إذا جعلته True فستحتفظ مجموعة البيانات الأصليّة بالنسخة الحالية للسجلات، وسيتم المزج فقط على مستوى النسخة الأصلية... دعنا نفهم هذا بمثال صغير: افترض أن لدينا سجلا في مجموعة البيانات، فيه خانة قيمتها الأصلية ١، وقيمتها الحالية ٢.. نريد أن نمزج هذا السجل بسجل مماثل له، لكن القيمة الأصلية لهذه الخانة فيه هي ٣، وقيمتها الحالية هي ٤.. لو كانت قيمة هذا المعامل True، فستصير القيمة الأصلية لهذه الخانة في مجموعة البيانات بعد المزج ٣، لكن ستظل قيمتها الحالية ٢.. أما إذا جعلت قيمتها False، فستصير القيمة الأصلية لهذه الخانة في مجموعة البيانات بعد المزج ٣، وقيمتها الحالية ٤.. الجدول التالي يلخص هذا المثال:

| | القيمة الأصلية | القيمة الحالية |
|---|---|---|
| سجل مجموعة البيانات | ١ | ٢ |
| السجل الممزوج | ٣ | ٤ |
| سجل مجموعة البيانات بعد المزج (قيمة المعامل True) | ٣ | ٢ |
| سجل مجموعة البيانات بعد المزج (قيمة المعامل False) | ٣ | ٤ |

لاحظ أن جعل هذا المعامل True، هو الطريقة الوحيدة التي تستطيع بها تغيير القيمة الأصلية دون تغيير القيمة الحالية، لأن صيغ الوسيلة

DataRow.Item التي تتيح لك تحديد النسخة التي تتعامل معها، قابلة للقراءة فقط، ولا يمكن استخدامها للكتابة!

وقد استخدمنا الوسيلة Merge في المشروع OptimisticConcurrency مرتين:

- مرة في حدث ضغط القائمة الموضعية "أريد حفظ تعديلاتي"، وقد أرسلنا إلى المعامل الثاني لهذه الوسيلة القيمة True لتغيير النسخة الأصلية للسجل المراد إعادة حفظه، مع الاحتفاظ بتغييرات المستخدم لحفظها في قاعدة البيانات.
- ومرة في حدث ضغط القائمة الموضعية "إلغاء تعديلاتي"، وقد أرسلنا إلى المعامل الثاني لهذه الوسيلة القيمة False للتخلص من السجل القديم، ووضع السجل القادم من قاعدة البيانات بدلا منه (يشمل هذا النسخة الأصلية والنسخة الحالية للسجل).

٣- بعض الصيغ تزيد على الصيغ السابقة بمعامل ثالث، يحدد ردّ الفعل الذي سيتخذ لو كان تركيب مجموعتي البيانات مختلفا (كعدم وجود بعض الجداول أو الأعمدة في مجموعة البيانات الحالية)، وهو يأخذ إحدى قيم المرقم MissingSchemaAction التي تعرفنا عليها من قبل.. تذكر أن القيمة الافتراضية في الصيغ التي لا تستقبل هذا المعامل هي Add، بمعنى إضافة الجداول والأعمدة اللازمة إلى مجموعة البيانات الحالية لاستقبال البيانات الجديدة من مجموعة البيانات المضافة.

ولا يتمّ التحقق من صحّة القيود Constrains، إلا بعد اكتمال عمليّة المزج.. فإذا كانت هناك سجلات تعارض القيود المفروضة، يحدث ما يلي:

- ينطلق خطأ في البرنامج من النوع ConstraintException.
- توضع القيمة False في الخاصية DataSet.EnforceConstraints لإيقاف تطبيق القيود، وذلك حتى يمكن الاحتفاظ بالبيانات الممزوجة إلى أن ترى كيف تحل المشكلة.
- يوضع نص الخطأ في الخاصية RowError الخاصة بكل سجل يتعارض مع القيود المفروضة، لهذا عليك فحص هذه الأخطاء وإصلاحها بالطريقة المناسبة، قبل محاولة وضع القيمة True في الخاصية EnforceConstraints من جديد لتطبيق القيود.

- **تخمين المخطط InferXmlSchema:**

تقرأ كود XML، وتحاول استنتاج مخططات الجداول من بيانات السجلات الموجودة فيها، وتحمل هذا المخطط في مجموعة البيانات.. ولهذه الوسيلة عدة صيغ، كل منها لها معاملان:

- المعامل الأول يحدد الملف الذي يوجد به كود XML، سواء كان ذلك في صورة مسار الملف، أو كائن مجرى بيانات Stream، أو قارئ نصي TextReader، أو "قارئ XML" XmlReader.
- المعامل الثاني يستقبل مصفوفة نصية، تحتوي على أسماء عناوين المواقع Url التي تريد استبعادها عند استخلاص المخطط من الملف.

- **الحصول على كود المخطط GetXmlSchema:**

تعيد نصا يحتوي على كود XML الذي يمثل مخطط الجداول الموجودة في مجموعة البيانات.

- **الحصول على الكود GetXml:**

تعيد نصا يحتوي على كود XML الذي يمثل البيانات الموجودة في مجموعة البيانات.

- **كتابة كود المخطط WriteXmlSchema:**

تحفظ كود XML الذي يمثل مخطط جداول مجموعة البيانات، في الملف المرسل إليها كمعامل، سواء كان في صورة مسار الملف، أو كائن مجرى بيانات Stream، أو قارئ نصي TextReader، أو "قارئ XML" XmlReader.

وتوجد عدة صيغ لهذه الوسيلة تزيد على الصيغ السابقة بمعامل ثان من نوع المندوب Converter(Of Type, String)، وهو يستقبل عنوان أي دالة لها معامل من النوع Type وتعيد String.. هذا مفيد إذا كانت مجموعة البيانات تحتوي على عمود يتعامل مع نوع بيانات مركب لا يمكن تحويله إلى نص مباشرة، وفي هذه الحالة يمكنك كتابة دالة مناسبة توضح كيف يمكن تحويل بياناته إلى نص، وترسلها إلى هذا المعامل.

- **كتابة الكود WriteXml:**

مماثلة للوسيلة السابقة، إلا أنها تحفظ سجلات مجموعة البيانات في ملف XML.. وهناك صيغة أخرى لهذه الوسيلة، لها معامل ثان من نوع المرقم XmlWriteMode الذي يمتلك القيم التالية:

| | |
|---|---|
| IgnoreSchema | كتابة السجلات فقط بدون كتابة مخطط البيانات.. هذه هي القيمة الافتراضية. |
| WriteSchema | كتابة السجلات ومخطط البيانات معا في الملف. |
| DiffGram | كتابة كل محتويات مجموعة البيانات في الملف، بما في ذلك النسخة الأصلية Original Version والحالية Current Version لكل السجلات، حتى لو لم تتغير النسخة الحالية للسجل عن النسخة الأصلية. |

وقد استخدمنا هذه الوسيلة في الزر "حفظ البيانات في ملف" في المشروع DataSetSample، وأرسلنا إلى المعامل الثاني القيمة WriteSchema لحفظ المخطط مع البيانات.. هذا يضمن لنا حفظ العلاقة بين الجدولين والقيود المفروضة عليهما، والمفاتيح الأساسية والفرعية.

| ملحوظة: |
|---|
| إذا أردت حفظ السجلات التي تغيرت فقط، فعليك استخدام الوسيلة DataSet.GetChanges للحصول على مجموعة بيانات جديدة بها السجلات التي تغيرت فقط، واستخدام الوسيلة WriteXml الخاصة بهذه المجموعة الجديدة لحفظ سجلاتها. |

### قراءة كود المخطط ReadXmlSchema:

مماثلة للوسيلة WriteXmlSchema في معاملاتها، ولكنها تقوم بالوظيفة العكسية، حيث تقرأ المخطط من ملف XML وتحمله في مجموعة البيانات.. لاحظ أن هذه الوسيلة قد تتسبب في حدوث أخطاء إذا كانت مجموعة البيانات تحتوي على مخطط بالفعل، لهذا عليك استدعاء الوسيلة DataSet.Reset أولا لمحو كل بياناتها ومخططاتها أولا، قبل استدعاء الوسيلة ReadXmlSchema.

### قراءة الكود ReadXml:

مماثلة للوسيلة السابقة، إلا أنها تقرأ بيانات السجلات من ملف XML وتحملها في مجموعة البيانات.. وهناك صيغة أخرى لهذه الوسيلة، لها معامل ثانٍ من نوع المرقم XmlReadMode الذي يمتلك القيم التالية:

| | |
|---|---|
| Auto | القيمة الافتراضية. |
| Read Schema | قراءة السجلات، وقراءة المخطط إن وجد في الملف (يجب أن ترسل إلى المعامل الثاني للوسيلة WriteXml القيمة WriteSchema ليتم حفظ المخطط مع البيانات، وبالتالي يمكنك قراءته).. وإذا كان بمجموعة البيانات مخطط بالفعل، تتم إضافة الجداول الجديدة إليه، لكن خطأ سيحدث لو كانت مجموعة البيانات تحتوي على جدول له نفس اسم جدول موجود في المخطط.<br>ويؤدي طلب قراءة المخطط من ملف يحتوي على البيانات فقط، إلى عدم تحميل أي منهما في مجموعة البيانات! |
| Ignore Schema | قراءة السجلات فقط، مع تجاهل أي مخطط موجود. |
| Infer Schema | تتجاهل أي مخطط في الملف، وتحاول استنتاج المخطط من بيانات السجلات، وتضيف المخطط والسجلات إلى مجموعة البيانات.. ويحدث خطأ إذا كانت مجموعة البيانات تحتوي على مخطط بالفعل، وكان يحتوي على عمود تتعارض تفاصيله مع عمود موجود في المخطط المضاف. |
| Infer Typed Schema | مماثلة للقيمة السابقة، إلا أنها تستنتج نوع بيانات كل عمود، فإن فشلت تعتبر أن نوع العمود String. |
| Diff Gram | تقرأ السجلات الأصلية والحالية من الملف، وذلك إذا كنت حفظتها فيه سابقا باستخدام القيمة DiffGram.. وإذا كانت مجموعة البيانات تحتوي على سجلات بالفعل فستحتفظ بها، وستضاف إليها السجلات الجديدة. |
| Fragment | استخدم هذه القيمة إذا كان الملف يحتوي على أجزاء من كود XML وليس كود وثيقة كاملة. |

وقد استخدمنا هذه الوسيلة في الزر "قراءة البيانات من ملف" في المشروع DataSetSample، وأرسلنا إلى المعامل الثاني القيمة ReadSchema لقراءة المخطط مع البيانات.. هذا يضمن لنا إنشاء العلاقة بين الجدولين في مجموعة البيانات، لأن وظيفة البرنامج تحتاجها.

لاحظ أن الوسيلة ReadXml لا تستدعي الوسيلة AcceptChanges تلقائيا كما تفعل الوسيلة DataAdapter.Fill، لهذا فإن السجلات التي يتم تحميلها في مجموعة البيانات ستعتبر سجلات جديدة Addedd، ولو ضغطت زر الحفظ في قاعدة البيانات، فسيتم إضافة كل هذه السجلات مرة أخرى إلى

جدول المؤلفين وجدول الكتب، وهذا سيجعل البيانات مكررة!.. ولحل هذه المشكلة، عليك استدعاء الوسيلة AcceptChanges مباشرة بعد تحميل السجلات إلى مجموعة البيانات، وبهذا يتم اعتبار أنها لم تتغير، ولا يتم حفظها في مجموعة البيانات.

لكنك قد تريد اعتبار السجلات جديدة في بعض المواقف، وذلك إذا كنت تملك البيانات في ملف XML وتريد إضافتها إلى قاعدة بيانات فارغة.

وهناك ملاحظة بسيطة أخرى، وهي أن هذه الوسيلة لا يهمها امتداد الملف، بل يهمها فقط صحة محتوياته.. لهذا فقد أعطينا للملفات الخاصة بنا في المشروع CustomDataSet الامتداد dsf.، وهي امتداد من اختراعنا (اختصار للتعبير DataSet Format)، وجعلنا مربع حوار فتح ملف لا يعرض سوى الملفات التي لها هذا الامتداد، وبهذا نضمن أن الملفات التي نحاول قراءتها سيكون لها الصيغة المناسبة لمجموعة البيانات، فملفات XML تستطيع حمل أي نوع من البيانات وبأي تنسيق، لكنها لن تكون جميعا صالحة للعرض في برنامجنا.

وتمتلك مجموعة البيانات الحدث التالي:

### فشل الدمج MergeFailed:

ينطلق إذا فشلت عمليّة دمج بيانات جدولين باستخدام الوسيلة Merge.. يحدث هذا مثلا، إذا كان العمود المستخدم كمفتاح أساسي في السجل القادم، مختلفا عن العمود المستخدم كمفتاح أساسي في السجل الموجود مجموعة البيانات.

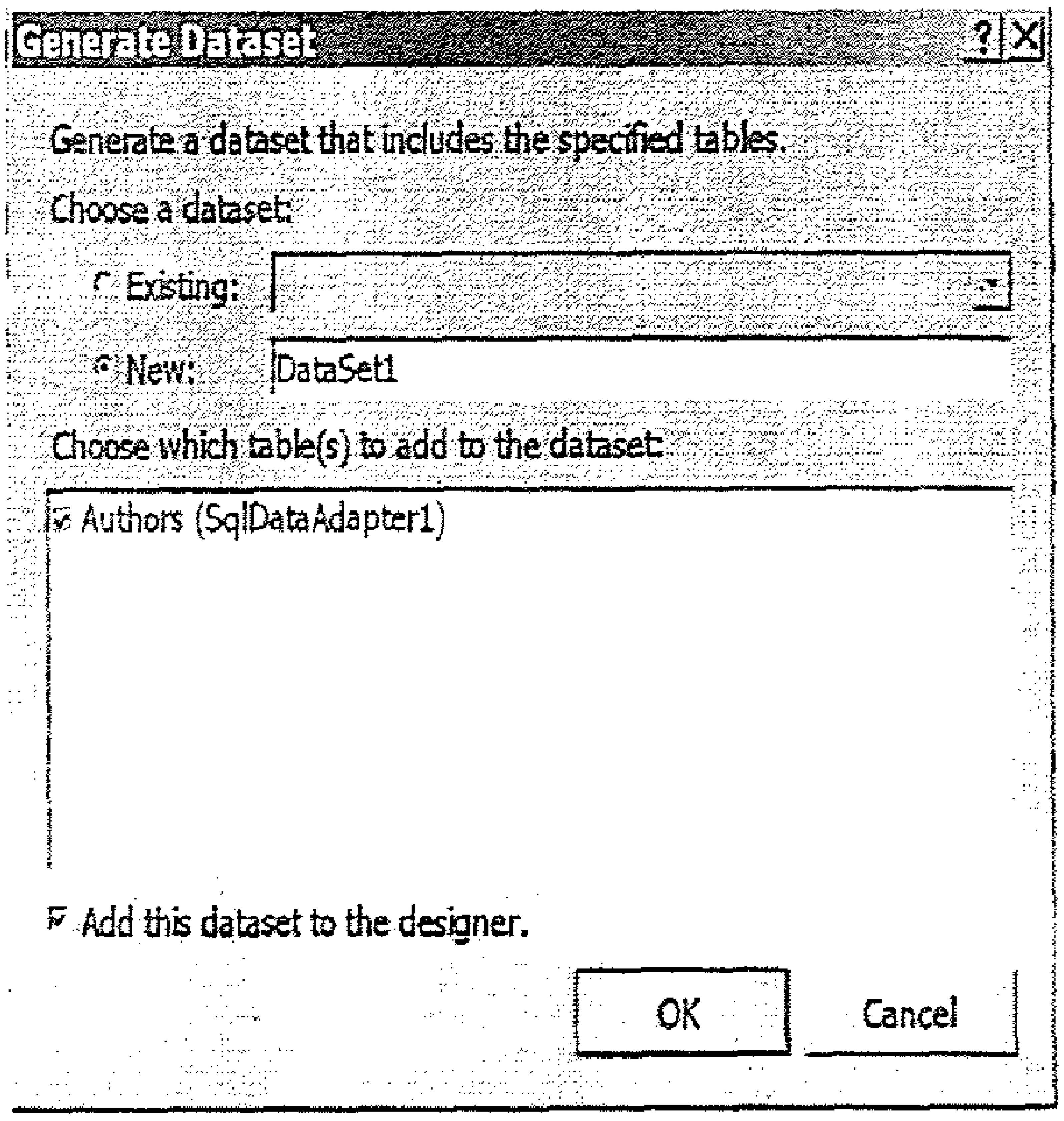
والمعامل الثاني e لهذا الحدث من النوع MergeFailedEventArgs، وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| | Table | تعيد كائن الجدول DataTable الذي رفض عملية الدمج. |
| | Conflict | تعيد نصا يشرح سبب التعارض الذي أدى إلى فشل عملية الدمج. |

## المعالج السحري لإنشاء مجموعة البيانات
## Generate DataSet Wezard

تتيح لك دوت نت طريقة مرئية لإنشاء مجموعة البيانات آليا.. لفعل هذا، أضف موصل بيانات Data Adapter إلى صينية مكونات النموذج، واضبط خصائصه كما تعلمنا من قبل، ثم اضغطه بزرّ الفأرة الأيمن، ومن القائمة الموضعيّة اضغط الأمر "إنتاج مجموعة البيانات" Generate Dataset.. وستجد نفس الأمر في القائمة الرئيسيّة Data أعلى النافذة.

سيظهر لك مربّع حوار "إنتاج مجموعة البيانات" كما هو موضح بالصورة:




في هذه النافذة يمكنك اختيار إنشاء مجموعة بيانات من مخطط مجموعة بيانات موجود سابقا في البرنامج، أو إنشاء مخطط جديد اسمه DataSet1.. لاحظ أنك تستطيع تغيير هذا الاسم، والأفضل اختيار اسم أكثر تعبيرا عن وظيفة مجموعة البيانات.

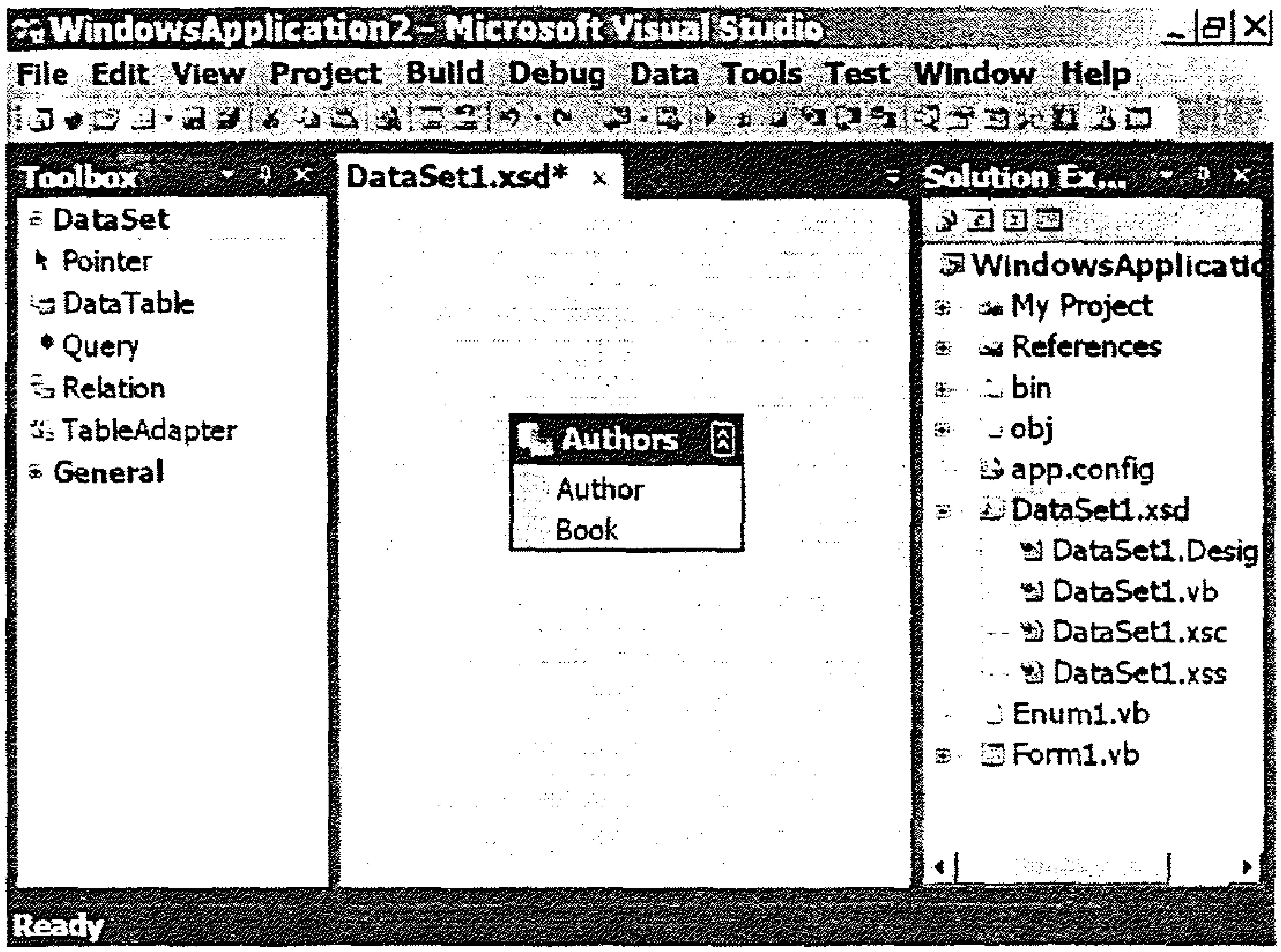
وتعرض لك النافذة قائمة بأسماء الجداول التي يوفرها موصل البيانات، ليمكنك اختيار إضافتها جميعا إلى مجموعة البيانات أو حذف بعضها.

ويوجد اختيار أسفل النافذة، يحدد إذا كنت تريد إضافة نسخة من مجموعة البيانات إلى النموذج أم لا.

بعد أن تحدد اختياراتك اضغط Ok لإغلاق النافذة.. سيؤدي هذا إلى ما يلي:

- إضافة ملفّ اسمه DataSet1.xsd إلى ملفات المشروع التي يعرضها متصفح المشاريع Solution Explorer.. والامتداد xsd هو اختصار للتعبير "لغة تعريف المخطط" Xml Schema Definition، لهذا لو فتحت هذا الملف من مجلد المشروع باستخدام برنامج Notepad، فستجده يحتوي على كود XML الذي يعرف مخطط مجموعة البيانات (الجداول والأعمدة والعلاقات والقيود التي تحتويها).. أما لو نقرت هذا الملف مرتين بالفأرة في متصفح المشاريع، فستعرض لك دوت نت نافذة

مصمم المخطط Schema Designer، وستجد فيها رسما مبسطا يمثل الجداول والأعمدة الموجودة في المخطط، كما هو موضح بالصورة:




- إنشاء فئة خاصة اسمها DataSet1 ترث فئة مجموعة البيانات DataSet Class.. كود هذه الفئة يوضع في الملف DataSet1.Designer.vb، والذي ستجده في العناصر الفرعية للمخطط DataSet1.xsd إذا عرضت كل ملفات المشروع بضغط الزر Show All Files الموجود أعلى متصفح المشاريع.

وتسمى الفئة DataSet1 بمجموعة البيانات محددة النوع Typed DataSet، وسنتعرف بعد قليل على معنى هذا المسمى وفائدته.

- إضافة مجموعة بيانات اسمها DataSet11 إلى صينية مكونات النموذج Component Tray.. هذه المجموعة هي نسخة معرفة من الفئة DataSet1، وستجد جملة تعريفها في ملف خصائص النموذج كالتالي:

**Friend WithEvents DataSet11 As DataSet1**

لاحظ أن الاسم الافتراضي DataSet11 يشير إلى أن هذه هي النسخة رقم ١ من الفئة dataSet1.. ولو كنت سميت مجموعة البيانات منذ البداية DsAuthorBooks مثلا، لكان اسم هذه النسخة هو DsAuthorBooks1 بدلا من DataSet11.

- ظهور أداة جديدة اسمها DataSet1 في أعلى صندوق الأدوات Toolbox تحت شريط خاص يحمل الاسم:

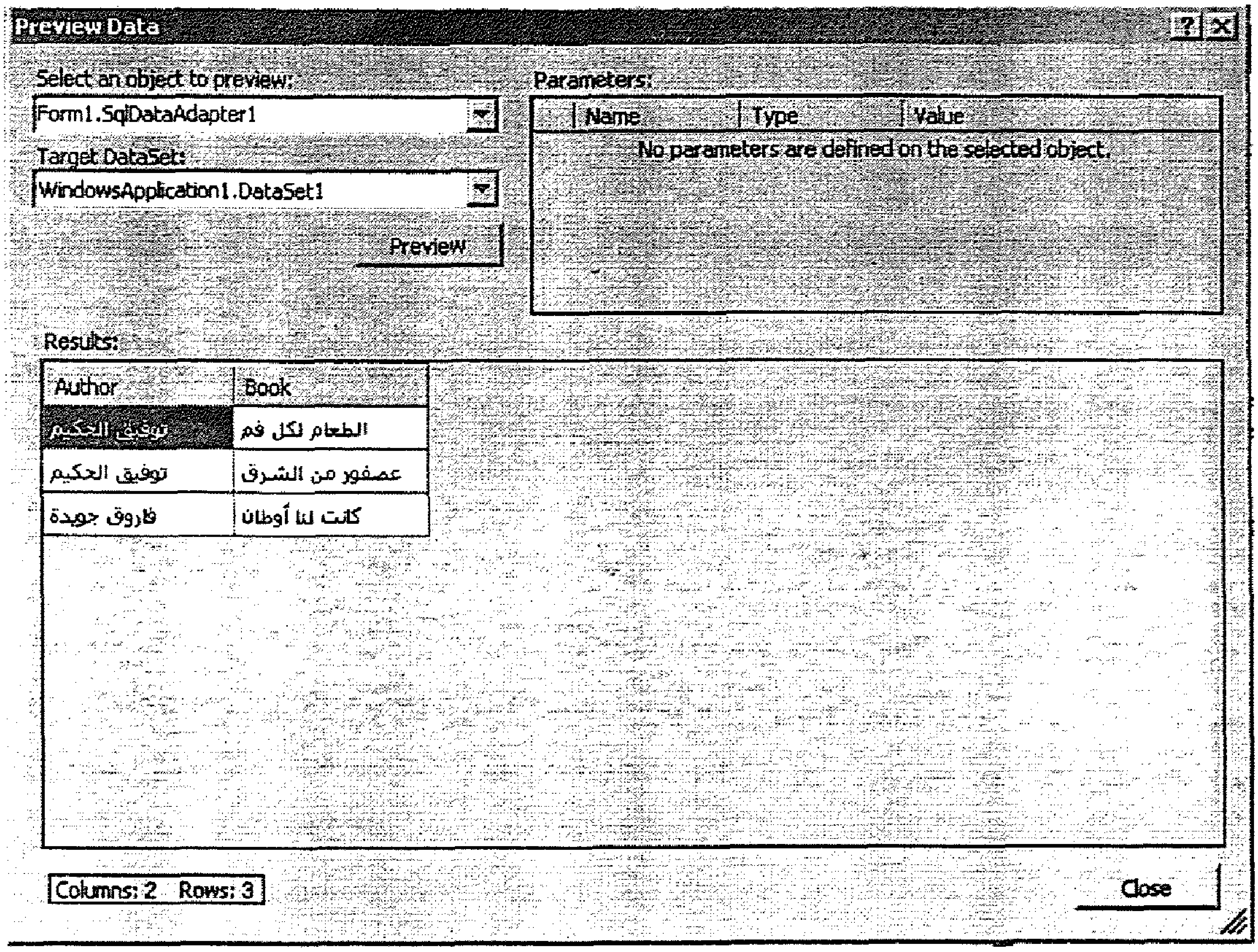
ProjectName Components

حيث ProjectName هو اسم المشروع.

وبهذا تستطيع إضافة نسخ منها إلى النموذج بطريقة مرئية.

والمشروع TypedDataSet يريك مثالا على هذا.

دعنا نرى ماذا فعلنا حتى هذه اللحظة.. من القائمة الرئيسيّة Data، اضغط الأمر Preview Data (ستجد هذا الأمر أيضا في القائمة الموضعيّة عند ضغط موصل البيانات في صينيّة المكوّنات بزر الفأرة الأيمن).. هذا الأمر سيفتح نافذة استعراض البيانات التالية:




أعلى يسار هذه النافذة، توجد قائمتان منسدلتان، تتيحان لك تحديد موصل البيانات ومجموعة البيانات التي تريد استعراض بياناتهما، وأعلى اليمين ستجد جدولا يعرض المعاملات التي تم تعريفها في أوامر التحديد والتحديث إن وجدت.. أما الجزء السفلي من النافذة، فيعرض السجلات الناتجة من تنفيذ أمر التحديد، وهو سيكون فارغة مبدئيّا، إلى أن تضغط الزر Preview.

## مجموعة البيانات محددة النوع Types DataSet:

رأينا كيف قامت دوت نت بإنشاء فئة اسمها DataSet1 آليا اعتمادا على المخطط DataSet1.xsd.. وتسمى هذه الفئة باسم مجموعة البيانات محددة النوع Typed DataSet، وذلك لأنها تقوم بتعريف أنواع خاصة لجداول وصفوف مجموعة البيانات، وتسمح لك بالتعامل مع الجداول والأعمدة بأسمائها مباشرة.. ولكي يحدث هذا، تقوم هذه الفئة بتعريف العديد من العناصر.. ولو فتحت الملف DataSet1.Designer.vb في المشروع TypedDataSet فستجد فيه تعريف الفئة DataSet1، وستجد فيها العناصر التالية:

١- فئة خاصة لأنواع صفوف كل جدول في مجموعة البيانات.. هذه الفئات تحمل أسماء على الصيغة XRow، حيث X هو اسم الجدول.

وترث فئة الصف فئة صف البيانات الأم DataRow، وبداخل هذه الفئة يتم تعريف خاصية باسم كل عمود من أعمدة الجدول، تعيد قيمة الخانة الموجودة في هذا العمود في هذا الصف.. فمثلا، ستجد داخل الفئة DataSet1 فئة اسمها AuthorsRow تمثل صف البيانات في جدول المؤلفين، وستجد بداخلها خاصيتين هما: Author و Book تعيدان اسم المؤلف واسم الكتاب في الصف الحالي.

٢- فئة لكل جدول موجود في مجموعة البيانات.. هذه الفئات تحمل أسماء على الصيغة XDataTable، حيث X هو اسم الجدول.

وترث فئات الجداول الفئة عامة النوع TypedTableBase(Of T) والتي ترث بدورها فئة الجدول DataTable، حيث T هو نوع صفوف الجدول.

فمثلا، ستجد داخل الفئة DataSet1 فئة اسمها AuthorsDataTable تمثل جدول المؤلفين، وهي ترث الفئة TypedTableBase(Of AuthorsRow).

وبداخل فئة الجدول، يتم تعريف خصائص للتعامل مع كل عمود بالجدول، وهي تعيد كائنات من نوع فئة العمود DataColumn Class.. فمثلا، ستجد في الجدول AuthorsDataTable الخاصيتين AuthorColumn و BookColumn اللتين تتيحان لك التعامل مع عمودي المؤلفين والكتب.

كما يتم تعريف عدة أحداث لفئة الجدول إضافة إلى ما ترثه من الفئة DataTable، وهي: الصف يتغير XRowChanging، الصف تغير XRowChanged، الصف يحذف XRowDeleting، الصف حذف XRowDeleted.. حيث X هي اسم الجدول.. مثلا: في جدول المؤلفين يتم تعريف هذه الأحداث التالية: AuthorsRowChanging، AuthorsRowChanged، AuthorsRowDeleting، AuthorsRowDeleted.

٣- عدة خصائص على مستوى الفئة DataSet1 تحمل أسماء جداول مجموعة البيانات، لتتيح لك الحصول على كائن من نوع فئة هذا الجدول.. فمثلا، ستجد في الفئة DataSet1 خاصية اسمها Authors، تعيد نسخة من الفئة AuthorsDataTable، ويمكنك من خلالها التعامل مع جدول المؤلفين.

لكن لماذا كل هذا؟.. وبم تفيدنا المجموعة محددة النوع يا ترى؟
انظر مثلا إلى الجملة التالية، التي تقرأ اسم المؤلف الموجود في الصفّ الثالث في جدول المؤلفين:

**DataSet11.Tables("Authors").Rows(2).Item("Author")**

واضح طبعا أنها جملة طويلة تدفع إلى الاستياء.. فما رأيك إذن في الجملة التالية:

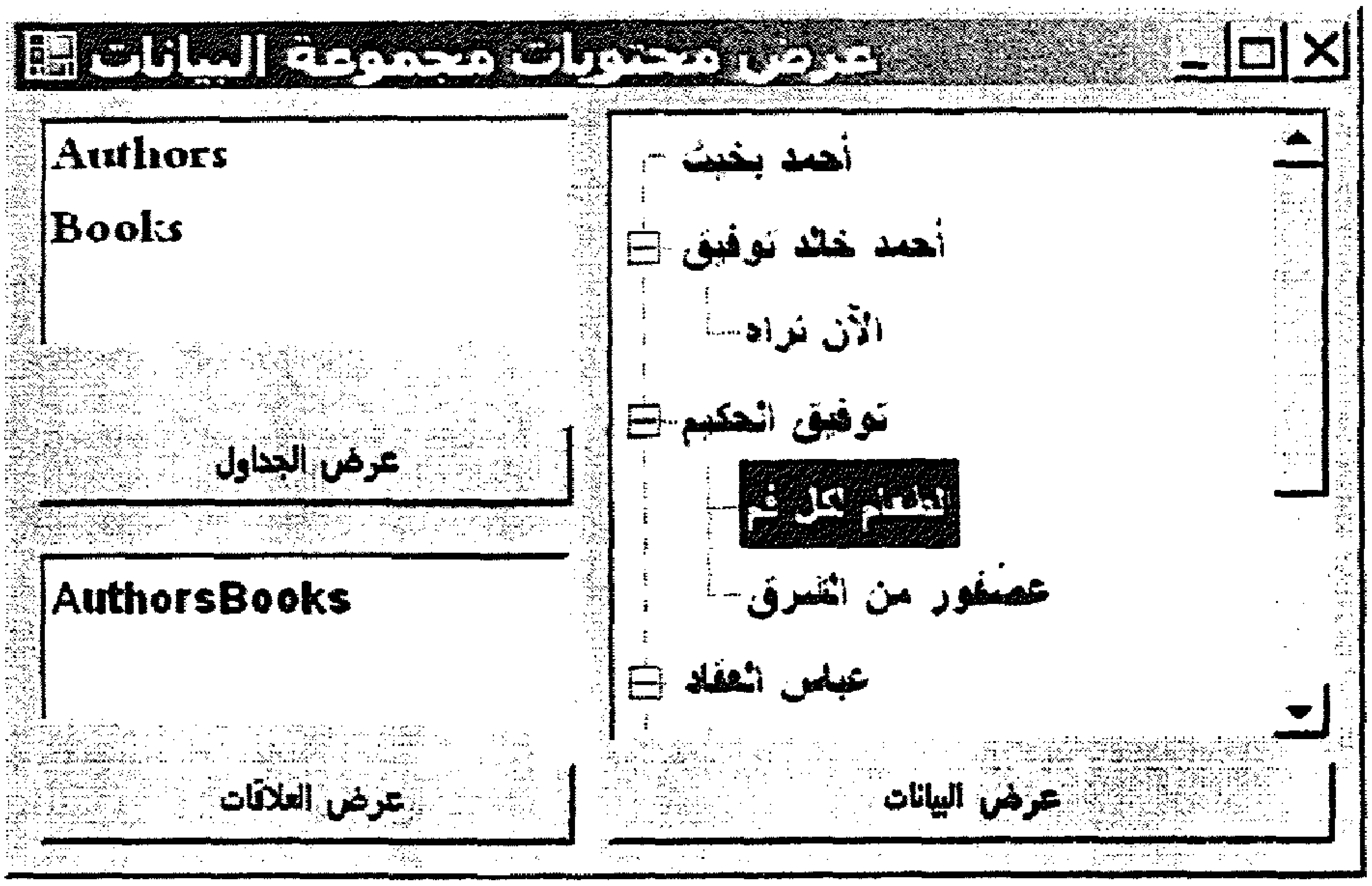
**DataSet11.Authors(2).Author**

إنّ الجملتين كلتيهما ـ ويا للعجب ـ متكافئتان، وإن كانت الأولى عامّة تستخدم خصائص فئة مجموعة البيانات الأم DataSet Class، بينما الثانية خاصّة، تستخدم خصائص مجموعة البيانات DataSet1.. لكن الجملة الثانية تمنحك الميزات التالية:

١- مختصرة وواضحة ومفهومة.
٢- أقل عرضة للخطأ.. ففي الجملة الأولى (الطويلة) هناك احتمالان للخطأ، وذلك أثناء كتابتك لاسمي الجدول Authors والعمود Author، لأنك تكتبهما يدويا كنصوص، ولا يتم اكتشاف أي خطأ فيهما إلا أثناء تشغيل البرنامج.. أما في الجملة الثانية (القصيرة)، فإنك تتعامل مع خصائص معرفة سابقا في الفئة DataSet1، ولن يقبل محرر الكود أي خطأ في أسمائها، مما يعني انعدام أي فرصة للخطأ.
٣- لا تحتاج عند كتابتها إلى تذكر أسماء الجداول والأعمدة بنفسك، وهو أمر تتضح أهميته في قواعد البيانات الضخمة التي تحتوي على عشرات الجداول، التي يحتوي كل منها على عشرات الأعمدة، مما يعني أنك ستضيع الكثير من الوقت لو استخدمت مجموعة بيانات عادية، لأنك ستضطر إلى العودة إلى قاعدة البيانات كثيرا لتذكر أسماء عناصرها.. بينما مجموعة البيانات محددة النوع تجعل الحياة جنة، لأن الاستشعار الذكي IntilliSense سيعرض لك قائمة الأسماء بمجرد كتابة النقطة . لتختار منها اسم الجدول أو العمود الذي تريد التعامل معه.

كل هذا يوضح لك فوائد مجموعة البيانات محددة النوع، وكيف تختصر وتسهل كتابة الكود بشكل كبير.

والمشروع DataSetContents يريك كيف يمكن عرض كل جداول وعلاقات وبيانات مجموعة البيانات محددة النوع، بالطريقة الموضحة في الصورة:




**ملحوظة:**

يمكنك استعارة مخطط XML من مشروع آخر، وإنشاء مجموعة بيانات محددة النوع بناء عليه.. لفعل هذا، اتبع الخطوات التالية:

- أنشئ مشروعا جديدا.
- من القائمة العلوية Project، اضغط الأمر Add Existing Item.
- استخدم مربع حوار فتح ملف للوصل إلى مجلد المشروع DataSetContents، واختر الملف DsAuthorsBooks.xsd.. سيضاف هذا الملف إلى المشروع.
- اعرض النموذج، وافتح صندوق الأدوات، وأضف مجموعة بيانات إلى النموذج.. وفي مربع الحوار الذي سيظهر اختر Typed DataSet.. ستجد أن القائمة المنسدلة تعرض العنصر X.DsAuthorsBooks، حيث X هو اسم المشروع.. اضغط OK.
- ستضاف مجموعة بيانات اسمها DsAuthorsBooks1 إلى صينية المكونات.. يمكنك استخدام نافذة الخصائص لتغيير اسمها إلى أي اسم مناسب، وليكن Ds.
- اضغط هذه المجموعة بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر EditIin DataSet Designer.. سيؤدي هذا إلى فتح مخطط XML، وستجد فيه مخطط جدول المؤلفين، ومخطط جدول الكتب، والعلاقة بينهما.

## إنشاء مجموعات بيانات خاصّة Custom DataSet:

في هذا المقطع سننشئ مجموعات بيانات بدون تحميل أية تفاصيل من قاعدة البيانات.. سننشئها باستخدام مخطط XML، وسنربطها بجدول عرض DataGridView بحيث يستطيع المستخدم إدخال البيانات بها، وسنسمح له بحفظ هذه البيانات في ملفّ XML، وإعادة تحميلها بعد ذلك كما يشاء.

ابدأ مشروعا جديدا اسمه CustomDataSet، ومن القائمة الرئيسية Project اضغط الأمر Add New Item لعرض نافذة إضافة عنصر.. من القائمة اليسرى اختر العنصر Data، ومن القائمة اليمنى اختر العنصر DataSet، وحدّد اسما لهذا العنصر الجديد وليكن MyDataSet، واضغط الزرّ OK.

سيضاف مخطّط XML إلى المشروع اسمه MyDataSet.xsd.. انقره مرتين بالفأرة لعرض مصمم المخطط.

لو فتحت صندوق الأدوات الآن، فستجد به أدوات تناسب مخطط XML، وستكون مبوبة تحت الشريط DataSet.. انقر مرّتين بالفأرة على العنصر DataTable لإضافة جدول جديد إلى المخطّط.. هذا الجدول سيظهر على المخطّط في صورة مستطيل فارغ، يحمل الاسم الافتراضيّ DataTable1.. لتغيّر هذا الاسم، اضغطه بالفأرة لإظهار مربع التحرير، واكتب الاسم الجديد Students، ثم اضغط Enter.. كما يمكنك استخدام نافذة الخصائص لتغيير اسم الجدول.

ولإضافة عمود إلى هذا الجدول، اضغطه بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط Add ثم Column.. حرر الاسم الافتراضي للعمود الجديد، واجعل اسمه ID.. اضغط F4 لعرض نافذة الخصائص، واستخدم الخاصية DataType لجعله من النوع Int32.. ويمكنك استخدام باقي الخصائص للتحكم في العمود بالطريقة التي تناسبك.. مثلا: اجعل للخاصية AutoIncrement القيمة True لجعل هذا الحقل ترقيما تلقائيا، ولا تنسَ أن تجعل للخاصيتين AutoIncrementSeed و AutoIncrementStep القيمة ١.

اضغط الهامش الأيسر للعمود ID بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Set Primary Key لجعله المفتاح الأساسي.

أضف إلى الجدول عمودا جديدا بنفس الطريقة واجعل اسمه Name.. سيكون نوع هذا العمود String بصور افتراضية، فاتركه كما هو.. يمكنك أن تحدد الخاصية Unique في نافذة الخصائص وتجعل قيمتها True، لتجعل اسم التلميذ متفردا غير قابل للتكرار.. ويمكنك أن تضع في الخاصية MaxLength القيمة ٣٠ لرفض أي اسم أطول من ٣٠ حرفا.

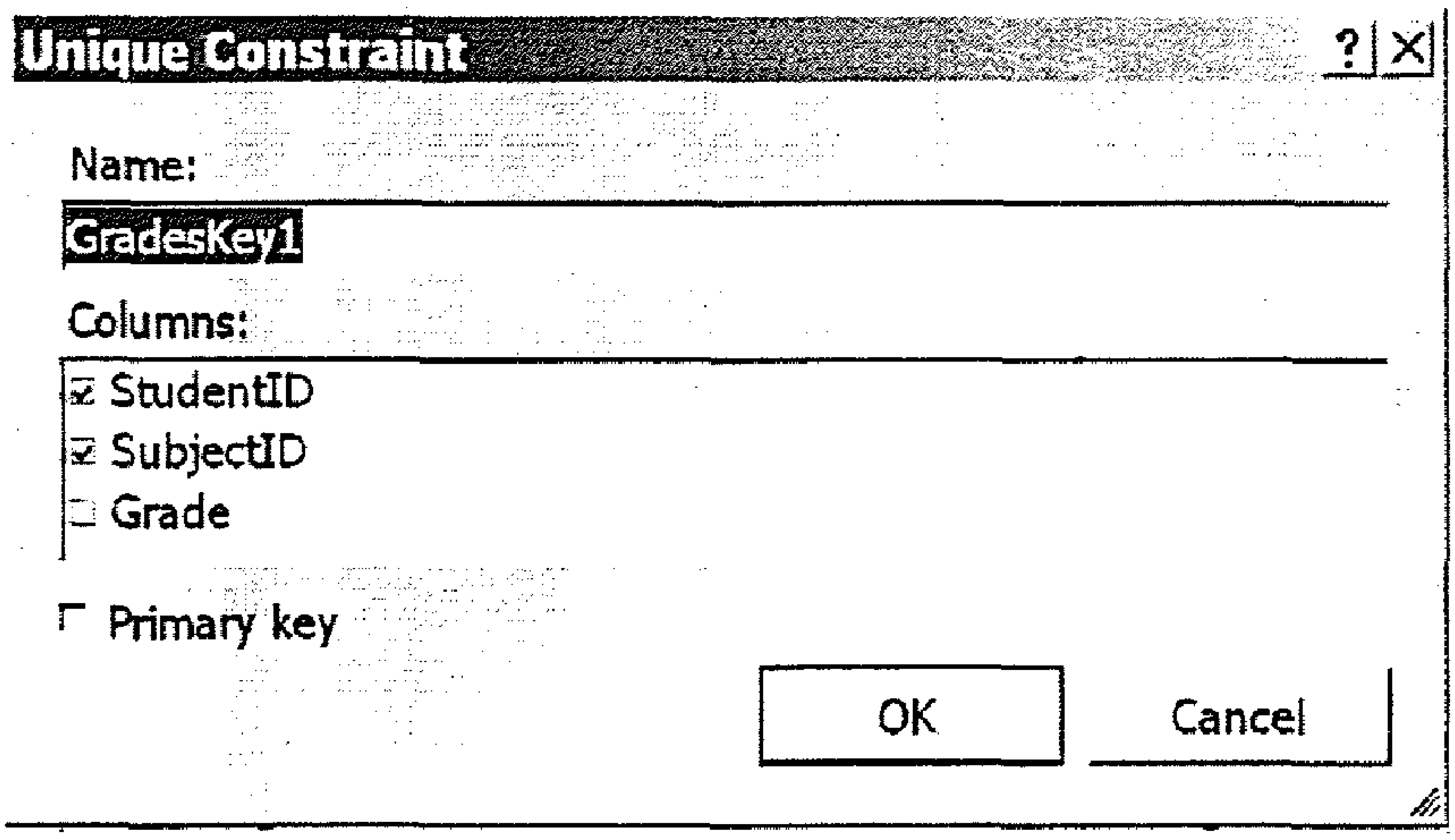
ولو أردت إدراج أي عمود قبل العمود Name، فاضغط هامشه الأيسر بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Insert Column.. وتستطيع حذف أي عمود في أي لحظة بتحديده وضغط الزر Delete.

بنفس الطريقة يمكنك إضافة جدول آخر اسمه Subjects، فيه العمودان: ID و Name.. لاحظ أنك تستطيع نسخ الجدول Students باستخدام الأمر Copy ولصق نسخة جديدة منه باستخدام الأمر Paste، حيث سيأخذ الجدول الجديد الاسم Students1، والذي يمكنك تغييره إلى Subjects.. هذا يسهل عليك إنشاء الجداول المتشابهة في تركيبها.

أضف جدولا ثالثا اسمه Grades، وأضف إليه الأعمدة StudentID و SubjectID و Grade، ولا تنس أن تغير نوع بياناتها جميعا إلى Int16.

واضح أننا سنسجل في الجدول Grades درجات كل طالب في كل المواد.. هذه علاقة متعدد بمتعدد Many-to-Many، فالطالب مرتبط بكل المواد، والمادة مرتبطة بكل الطلاب.. هذه فرصة لنجرب التعامل مع هذه العلاقة.

ويجب هنا أن نجعل الحقلين StudentID و SubjectID معا زوجا متفردا، حتى لا نكرر درجة نفس التلميذ في نفس المادة.. لفعل هذا، حدد هذين الحقلين (بضغطهما بزر الفأرة مع ضغط الزر Ctrl من لوحة المفاتيح)، ثم انقرهما بزر الفأرة الأيمن، ومن القائمة الفرعية اضغط Add Key.. ستظهر نافذة إضافة قيد التفرد Unique Constraint، كما هو موضح في الصورة:




في مربع النص العلوي اكتب اسم القيد، وفي القائمة السفلية تأكد أنك اخترت الأعمدة التي سيتم تطبيق القيد عليها (ستجد العمودين StudentID و SubjectID مختارين فعلا لأنك حددتهما قبل فتح النافذة).. ولو أردت جعل هذين العمودين مفتاحا أساسيا للجدول أيضا، فضع علامة الاختيار أمام الاختيار

Primary Key أسفل النافذة.. لكننا لا نحتاج إلى هذا هنا.. اضغط OK لإنشاء قيد التفرد.

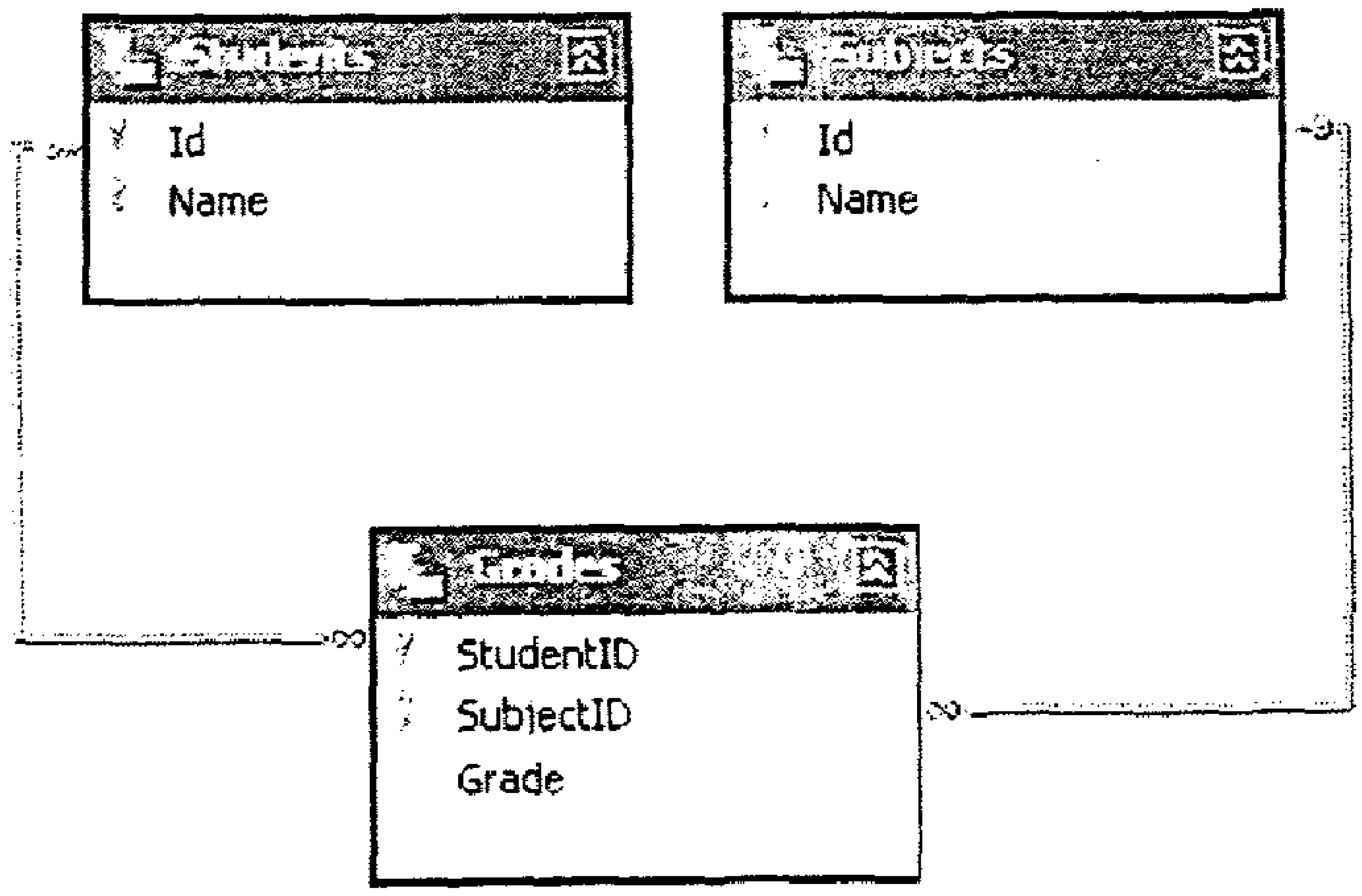
نريد الآن أن ننشئ العلاقات بين هذه الجداول.. يمكنك نقر العنصر Relation مرتين بالفأرة في صندوق الأدوات لعرض نافذة إنشاء العلاقة.. أو يمكنك أن تسحب العمود الأساسي من الجدول الرئيسي، وتسقطه على العمود الفرعي في الجدول التابع.. ولا تمر وأنت تسحب أي عمود على عمود آخر في نفس الجدول، وإلا فسيتم تحديده واعتباره جزءا من العلاقة.. لكنك تستطيع تصحيح ذلك في نافذة العلاقة على أي حال.. هذه النافذة مألوفة، ولن تجد فيها أي جديد لم نتعرف عليه سابقا.

نريد هنا أن نربط بين العمودين Students.ID و Grades.StudentID، وكذلك بين العمودين Subjects.ID و Grades.SubjectID.. ولا تنس المحافظة على التكامل المرجعي في كل علاقة، وذلك باختيار:

Both Relation And Foreign Key Constraint

من القسم: Choose what to create، على أن تجعل اختيارات الحذف والتحديث والرفض Cascade.. هذا سيريحنا من المشاكل التي تحدث عند حذف اسم طالب، أو تغيير اسم مادة، فالتكامل المرجعي سيحافظ على جدول الدرجات صحيحا دائما.

بعد إنجاز هذا، يجب أن يبدو المخطط كالتالي:




نريد الآن إنتاج مجموعة بيانات من هذا المخطط.. لفعل هذا انتقل إلى النموذج، وافتح صندوق الأدوات وانقر مرتين على العنصر DataSet.. سيظهر لك مربع حوار إضافة مجموعة بيانات، وستجد في القائمة العلوية اسم المخطط MyDataSet.. اضغط OK لإنتاج مجموعة بيانات محددة النوع من هذا المخطط، حيث ستضاف نسخة منها اسمها MyDataSet1 إلى صينية مكونات النموذج.. أقترح تغيير اسمها إلى DsStudents.

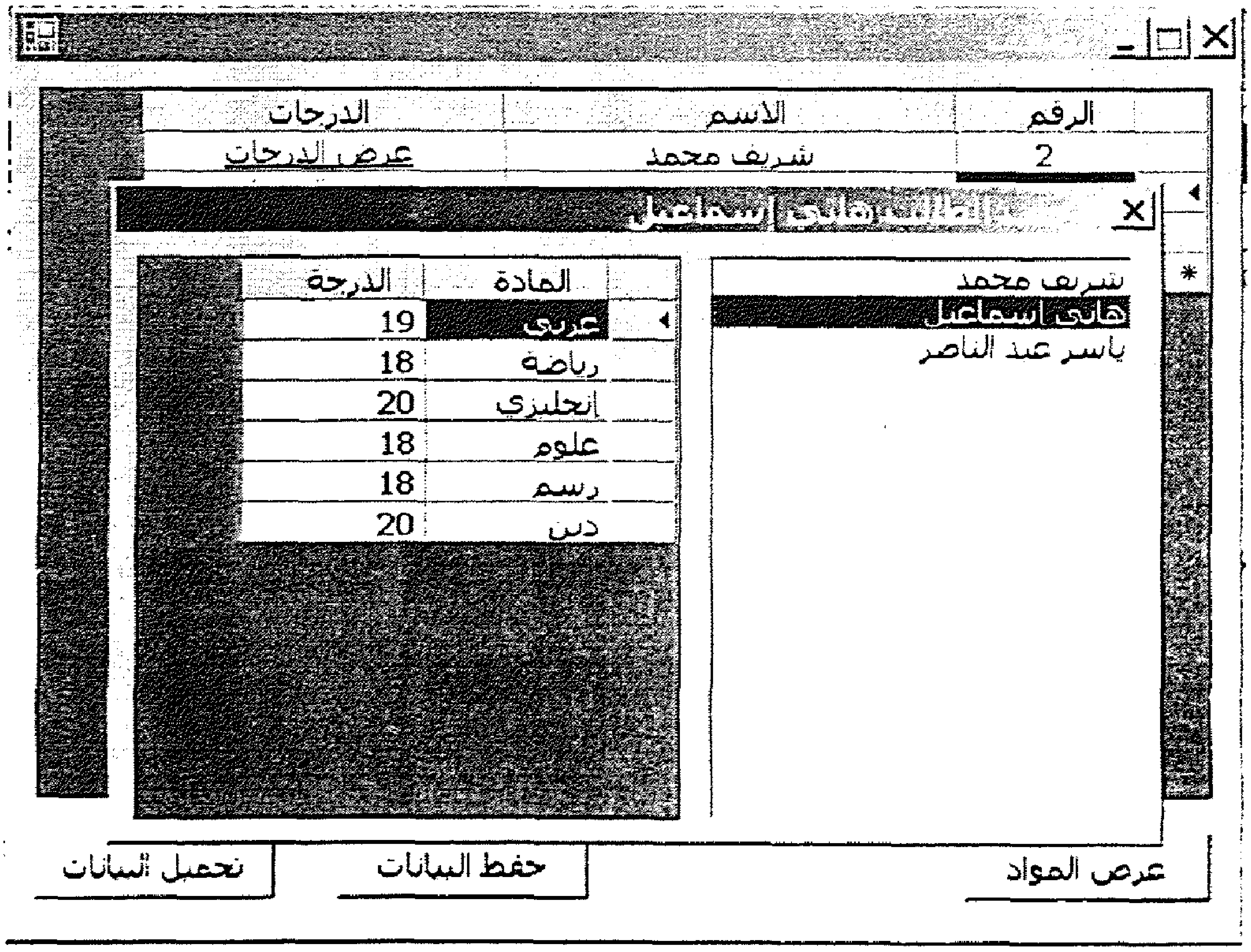
من هذه النقطة، يمكنك استخدام مجموعة البيانات بنفس الطريقة التي اعتدتها سابقا، وربطها بجدول العرض، وحفظ وتحميل البيانات بالطريقة المألوفة.. وستجد الكود الكامل الذي يفعل هذا في المشروع CustomDataSet.

لاحظ أن إجراء أي تعديل على مخطط XML، ينعكس مباشرة على فئة مجموعة البيانات محددة النوع، لهذا لست في حاجة إلى حذفها ثم إعادة إنشائها، فكل شيء يتم تلقائيا بمنتهى البساطة.

جرب مثلا إضافة عمود محسوب Calculated Column إلى الجدول Grades.. لفعل هذا افتح المخطط، واضغط الجدول Grades بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Insert Column.. سمِّ العمود الجديد Subject، وفي نافذة الخصائص حدد الخاصية Expression وضع فيها النص:

Parent(Subjects_Grades).Name

بمجرد أن تفعل هذا ستصير للخاصية ReadOnly القيمة True، وهذا معناه أن المستخدم لا يستطيع تعديل قيم هذا العمود، لأنه سيعرض ناتجا محسوبا بناء على قيمة عمود آخر.. وفي حالتنا هذه، جعلنا هذا العمود يعرض اسم المادة الدراسية، وذلك من خلال العلاقة Subjects_Grades التي تربط جدول المواد بجدول الدرجات، حيث سيستخدم العمود Subject قيمة الحقل الفرعي في هذه العلاقة (وهو الحقل SubjectID) ليحضر اسم المادة التي لها نفس الرقم من جدول المواد.. وسنتعرف على الأعمدة المحسوبة بتفصيل أكثر في فصل الجداول.



تلاحظ كما هو واضح في الصورة، إن إضافة اسم المادة إلى هذا الجدول سيجعلها تظهر في النافذة التي تعرض درجات الطالب، وهذا أفضل من إرباك المستخدم بعرض رقم المادة، والدرجة التي حصل عليها الطالب فيها.

لاحظ أننا استخدمنا قائمة ListBox لعرض أسماء الطلبة.. هذا يريح المستخدم أثناء إدخاله لدرجات الطلب، بسبب سرعة الانتقال من طالب إلى آخر.

ولعلك تتساءل: لمَ لا يدخل المستخدم أسماء الطلبة ودرجاتهم في نفس النموذج؟ هذا مجرد مثال مختصر، لكن في البرامج الحقيقية، ستحتاج إلى إدخال بيانات الطالب كاملة وليس اسمه فقط، مثل عمره، وفصله الدراسي، وعنوانه، وهاتفه... إلخ.. وكل هذا يحتاج إلى مساحة عرض كبيرة، والأفضل عمل نموذج مستقل له، وهو ما فعلناه في هذا المشروع، ليمكنك البناء عليه فيما بعد.

إلى الآن كل شيء رائع.. لكن تتبقى مشكلة في هذا البرنامج، وهي أن العمود المحسوب Subject سيتم حفظه في الملف عند حفظ مجموعة البيانات، وهو ما سيزيد من حجم الملف بلا ضرورة.. لهذا علينا حذف هذا العمود من مجموعة البيانات قبل حفظها.. لكن هذا سيؤدي إلى حدوث أخطاء في البرنامج!

ويمكن حل هذه المشكلة، باستخدام الوسيلة DataSet.Copy لنسخ مجموعة البيانات إلى مجموعة بيانات احتياطية، ثم حذف العمود Subject من هذه المجموعة الاحتياطية، وحفظها بياناتها في الملف.. وبهذا تظل مجموعة البيانات الأصلية كما هي، بينما نحصل على ملف أصغر حجما.. وعند تحميل هذا الملف، لن تحدث أية مشكلة في البرنامج بسبب غياب العمود Subject، فهو عمود محسوب، وسيستنتج البرنامج قيمته.. وستجد الكود الذي ينفذ هذا في الزر "حفظ البيانات".

لاحظ أن عيب هذه الطريقة هو أن نسخ مجموعة البيانات بكاملها قد يكون كارثة على الذاكرة إذا كان حجم بياناتها ضخما.. لهذا يمكن اللجوء إلى حل بديل، وهو حذف العمود من مجموعة البيانات قبل حفظها، ثم إنشائه مرة أخرى بعد الحفظ مباشرة.. أسهل طريقة لفعل هذا هي الاحتفاظ بمرجع للعمود في متغير من النوع DataCoulmn قبل حذفه من مجموعة أعمدة الجدول، ومن ثم حفظ البيانات في الملف، ثم إضافة العمود الذي نحتفظ بمرجعه إلى مجموعة الأعمدة مرة أخرى.. وستجد الكود الذي يفعل هذا في الزر "حفظ البيانات ٢".

**حفظ بيانات الشجرة في مجموعة البيانات:**

في المشروع CustomDataSet، رأينا مثالا على علاقة متعدد بمتعدد.. لعله يكون مناسبا الآن أن نرى مثالا على العلاقة الذاتية Self-Relation.. لفعل هذا، سننشئ مشروعا اسمه SaveTreeNodes، وهو يعرض شجرة ويتيح للمستخدم إضافة العناصر إليها، وتغيير مستوياتها، وهي وظائف تعلمنا كيف ننشئها في المشروع TreeViewSample في كتاب برمجة الويندوز، ولن نكرر شرحها هنا.. وستجد المشروع SaveTreeNodes ضمن أمثلة هذا الكتاب.

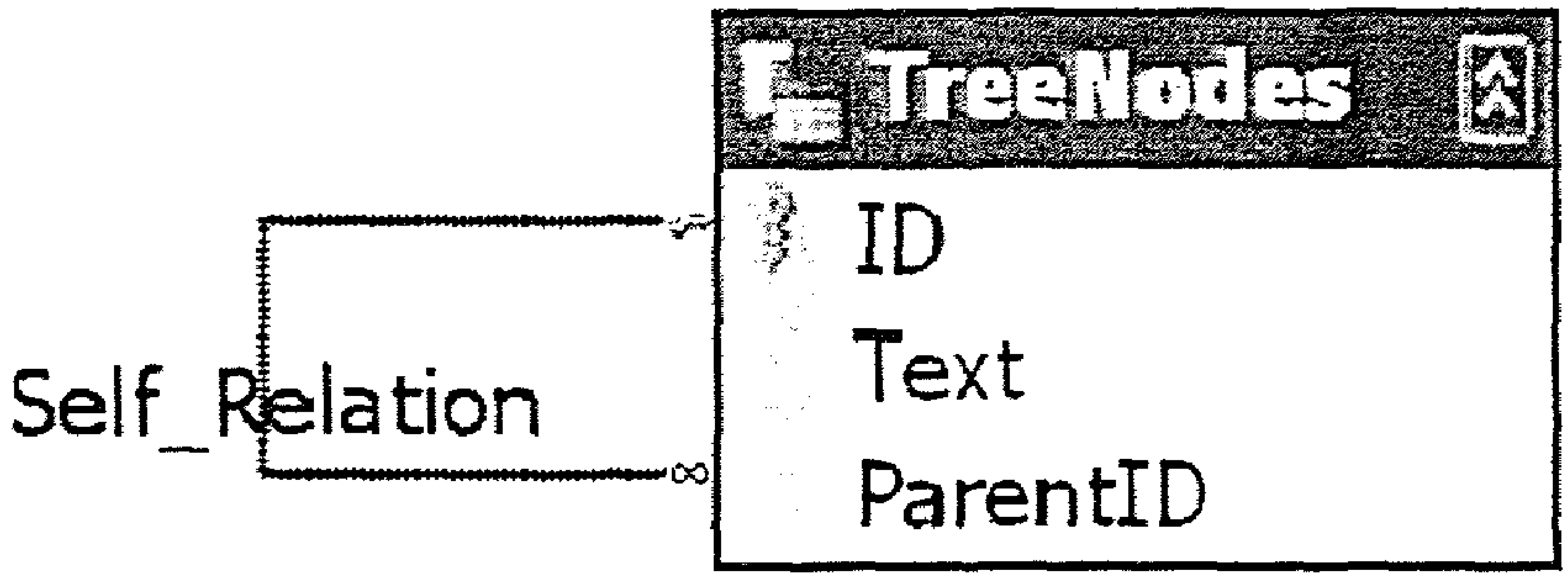
ما نريده الآن، هو أن نسمح للمستخدم بحفظ فروع الشجرة.. ونظرا، لأنه من غير العملي إنشاء قاعدة بيانات كاملة لحفظ بعض عناصر الشجرة، فسيكون من العملي هنا أن ننشئ مجموعة بيانات خاصة، ونستخدمها لحفظ العناصر في ملف XML.. وبهذا نكون قد استفدنا من قدرات مجموعة البيانات وعلاقاتها، وفي نفس الوقت سنحفظ البيانات في ملف مستقل.

لفعل هذا، أضفنا إلى المشروع SaveTreeNodes مخطط مجموعة بيانات بالطريقة المألوفة، وأسميناه TreeDataSet، وأضفنا إليه جدولا اسمه TreeNodes، وأضفنا إليه الأعمدة التالية:

| اسم العمود | نوع بياناته | وظيفته |
|---|---|---|
| ID | Int16 | المفتاح الأساسي، وهو يعمل أيضا كترقيم تلقائي. |
| Text | String | يحفظ نص فرع الشجرة. |
| ParentID | Int16 | المفتاح الفرعي، وهو يشير إلى رقم الفرع الرئيسي للفرع الحالي. |

وقد أضفنا علاقة إلى مخطط الجدول، لتربط بين الحقلين ID و ParentID.

هكذا يبدو شكل المخطط.. لاحظ كيف تخرج العلاقة من نفس الجدول وتعود إليه:



وقد أضفنا إلى النموذج نسخة من مجموعة البيانات وأسميناها TreeDs.. وحتى لا نعقد الأمور على أنفسنا، لن نملأ مجموعة البيانات بعناصر الشجرة إلا عند ضغط زر الحفظ، فهي مجرد وعاء وسيط يتيح لنا حفظ البيانات في ملف XML.. هذا هو كود هذا الزر:

```
TreeDs.Clear( )
For Each Node As TreeNode In TreeView1.Nodes
    Dim R = TreeDs.TreeNodes.AddTreeNodesRow(
            Node.Text, Nothing)
    SaveChildren(Node, R)
Next
TreeDs.WriteXml("C:\TreeNodes.Xml")
```

في البداية أفرغنا مجموعة البيانات من محتوياتها، ثم مررنا عبر جذور الشجرة لإضافتها إلى الجدول TreeNodes الموجود في مجوعة البيانات.. لاحظ أن مجموعة البيانات محددة النوع قد أضافت الوسيلة AddTreeNodesRow إلى الجدول، لتتيح لنا إضافة صف جديد إليه.. هذه الوسيلة تستقبل معاملين:

- نصا يمثل قيمة الحقل Text في الصف الجديد.
- كائن صف من النوع TreeDataSet.TreeNodesRow، لترسل إليه الصف الرئيسي للصف الحالي.. هذا أسهل من أن تضع بنفسك رقم الصف الرئيسي في الحقل ParentID، وهذه إحدى التسهيلات التي منحتها لك العلاقة الذاتية.. ونظرا لأن جذور الشجرة ليست لها فروع رئيسية، فسنرسل القيمة Nothing إلى هذا المعامل.. هذا سيترك الحقل ParentID فارغا.

بعد هذا، يجب أن نضيف إلى مجموعة البيانات فروع كل جذر.. لفعل هذا استخدمنا إجراء اسمه SaveChildren، وهو يستقبل معاملين:

- كائن الفرع TreeNode الذي سنضيف عناصره الفرعية إلى مجموعة البيانات.
- كائن الصف TreeDataSet.TreeNodesRow الذي يعمل كصف رئيسي، للصفوف التي سنضيفها الجدول.

هذا هو كود هذا الإجراء، مع ملاحظة أنه إجراء ارتدادي Recursive يستدعي نفسه، لأن كل فرع قد يحتوي على عناصر فرعية، كل منها قد يحتوي على عناصر فرعية، وهكذا:

```
Sub SaveChildren(ByVal ParentNode As TreeNode,
        ByVal ParentRow As TreeDataSet.TreeNodesRow)
    For Each Node As TreeNode In ParentNode.Nodes
        Dim R = TreeDs.TreeNodes.AddTreeNodesRow(
                Node.Text, ParentRow)
        SaveChildren(Node, R)
    Next
End Sub
```

وبعد وضع جميع بيانات الفروع في مجموعة البيانات، سيتم تنفيذ آخر سطر في كود ضغط زر الحفظ، وهو يستدعي الوسيلة WriteXML لحفظ محتويات مجموعة البيانات في الملف.

وهكذا نكون قد حفظنا فروع الشجرة بالكامل.. بقي إذن أن نعيد قراءتها من الملف عند ضغط زر التحميل.. لفعل هذا سنفرغ كلا من مجموعة البيانات والشجرة من محتوياتهما، ثم نقرأ بيانات الملف باستخدام الوسيلة ReadXML:

```
TreeDs.Clear( )
TreeView1.Nodes.Clear( )
TreeDs.ReadXml("C:\TreeNodes.Xml")
```

بعد هذا سننقل البيانات من مجموعة البيانات إلى الشجرة.. لفعل هذا سنضيف الجذور إلى الشجرة أولا.. نحن نعرف أن الجذر ممثل في الجدول بصف توجد في الخانة ParentID الخاصة به القيمة DbNull.. لهذا سنمر على كل الصفوف، ونستخدم الوسيلة الجاهزة IsParentIDNull التي عرفتها لنا مجموعة البيانات محددة النوع، لنرى إن كانت هذه الخانة فارغة، فإن كانت كذلك، عرفنا فرعا جديدا، ووضعنا فيه النص الموجود في الخانة Text، وأضفناه إلى الشجرة كجذر، ثم نستدعي الإجراء LoadChildren لتحميل العناصر الفرعية في هذا الجذر.. هذا هو الكود الذي يفعل هذا:

```
For Each R In TreeDs.TreeNodes
    If R.IsParentIDNull Then
        Dim Node = TreeView1.Nodes.Add(R.Text)
        LoadChildren(Node, R)
    End If
Next
```

أيضا، يجب أن يكون الإجراء LoadChildren ارتداديا Recursive يستدعي نفسه، لتحميل العناصر الفرعية لكل فرع في جميع المستويات.

ولكن كيف نعرف العناصر الفرعية؟

هذا سهل جدا، بفضل العلاقة الذاتية المعرفة في الجدول، فنحن نستطيع استخدام الوسيلة GetChildRows لمعرفة الصفوف الفرعية التابعة لأي صف رئيسي..

هذا هو كود هذا الإجراء:

```
Sub LoadChildren(ByVal ParentNode As TreeNode,
        ByVal ParentRow As TreeDataSet.TreeNodesRow)
    For Each R As TreeDataSet.TreeNodesRow In
        ParentRow.GetChildRows("Self_Relation")
          Dim Node = ParentNode.Nodes.Add(R.Text)
          LoadChildren(Node, R)
    Next
End Sub
```

هذا هو كل شيء.. يمكنك الآن تجربة البرنامج، وإضافة العناصر إلى الشجرة، وحفظها، ثم استرجاعها في أي وقت.

رائعة هي العلاقة الذاتية؟.. أليس كذلك؟

## فئة موصل الجدول TableAdapter Class

فئة موصل الجدول ليست فئة من فئات إطار العمل، لكنها فئة يقوم مصمم مجموعة البيانات محددة النوع Typed DataSet Designer بإنشائها في برنامجك لتسهيل تعاملك مع قواعد البيانات.

ويشبه موصل الجدول TableAdapter موصل البيانات DataAdapter في كل شيء، فهو يسمح لك بالحصول على السجلات من قاعدة البيانات لملء أحد جداول مجموعة البيانات.. ولكنه يتفوق على موصل البيانات في قدرته على تنفيذ عدد كبير من استعلامات التحديد SELECT للحصول على سجلات الجدول بطرق مختلفة من قاعدة البيانات، بشرط أن يكون الناتج ملائما لتركيب الجدول الذي يتم ملؤه.. بينما موصل البيانات مهيأ للتعامل مع استعلام واحد فقط.

ولكي تنشئ موصل جدول، يجب أن يحتوي برنامجك على مجموعة بيانات محددة النوع أولا.. اتبع هذه الخطوات:

- أنشئ مشروعا جديدا اسمه TableAdapter.
- من القائمة العلوية Project اضغط الأمر Add New Item.
- من القائمة اليسرى اختر العنصر Data، ومن القائمة اليمنى اختر العنصر DataSet، وامنحها الاسم DsAuthorsBooks.xsd.
- في نافذة المخطط، افتح صندوق الأدوات، واسحب العنصر TableAdapter وأسقطه على مصمم المخطط.. سيؤدي هذا إلى بدء المعالج السحري لتهيئة موصل الجدول:

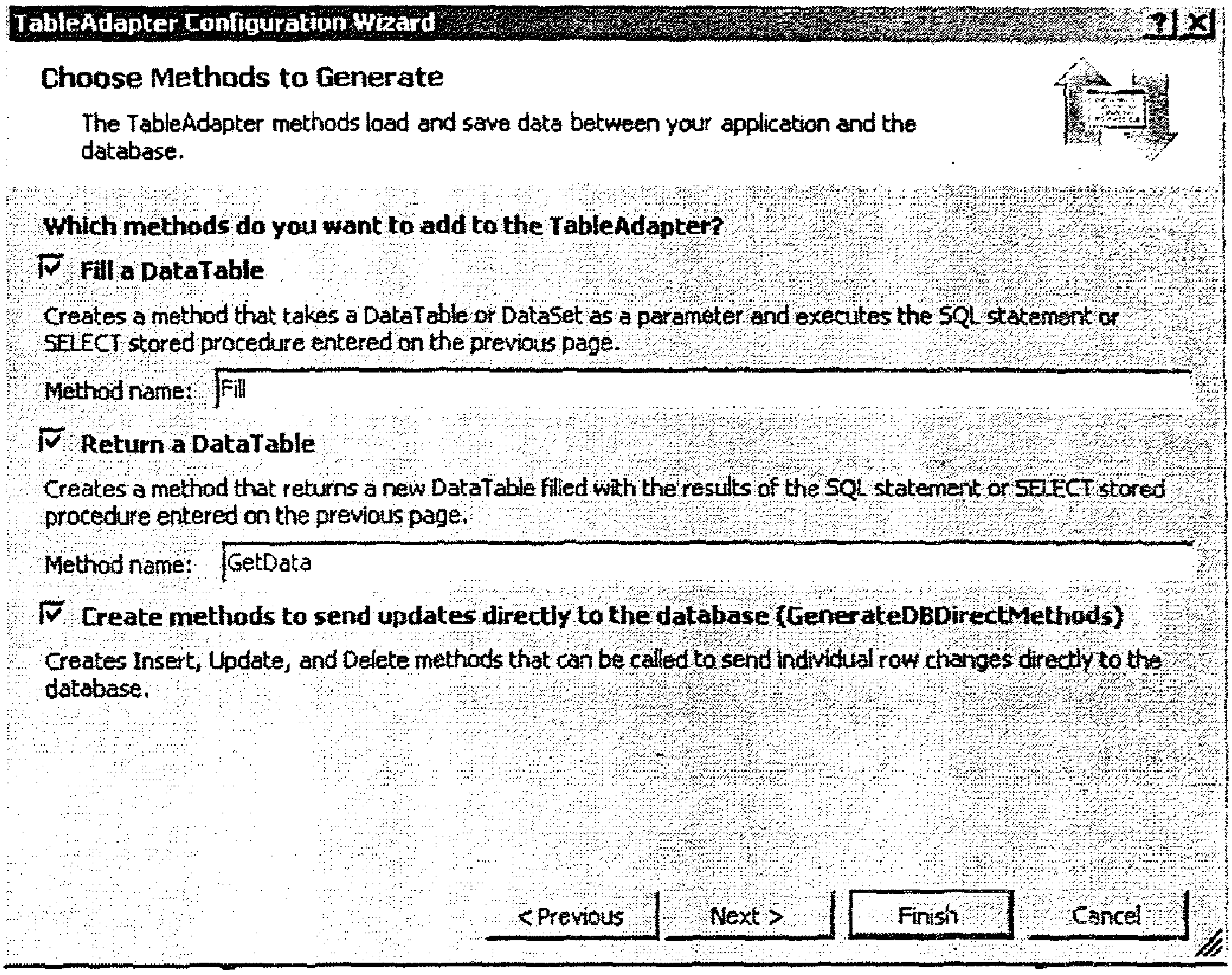
TableAdapter Configuration wizard.

- أول نافذة في هذا المعالج، هي نافذة اختيار قاعدة البيانات المراد الاتصال بها، وقد تعرفنا عليها من قبل.. اختر قاعدة بيانات الكتب Books.mdf من الاتصالات المتاحة، أو أنشئ اتصالا جديدا بها، ثم اضغط الزر Next.
- النافذة التالية ستسألك إن كنت تريد حفظ نص الاتصال في إعدادات البرنامج.. وافق على هذا ودع الاسم الافتراضي BooksConnectionString كما هو، واضغط Next.
- النافذة التالية تتيح لك اختيار نوع أمر التحديد.. اختر Use SQL Statement واضغط Next.
- في النافذة التالية اكتب استعلام التحديد التالي:

**SELECT * FROM Authors**

واضغط Next.

- في النافذة التالية يمكنك اختيار الوسائل التي ستضاف إلى فئة موصل الجدول، كما هو موضح في الصورة:



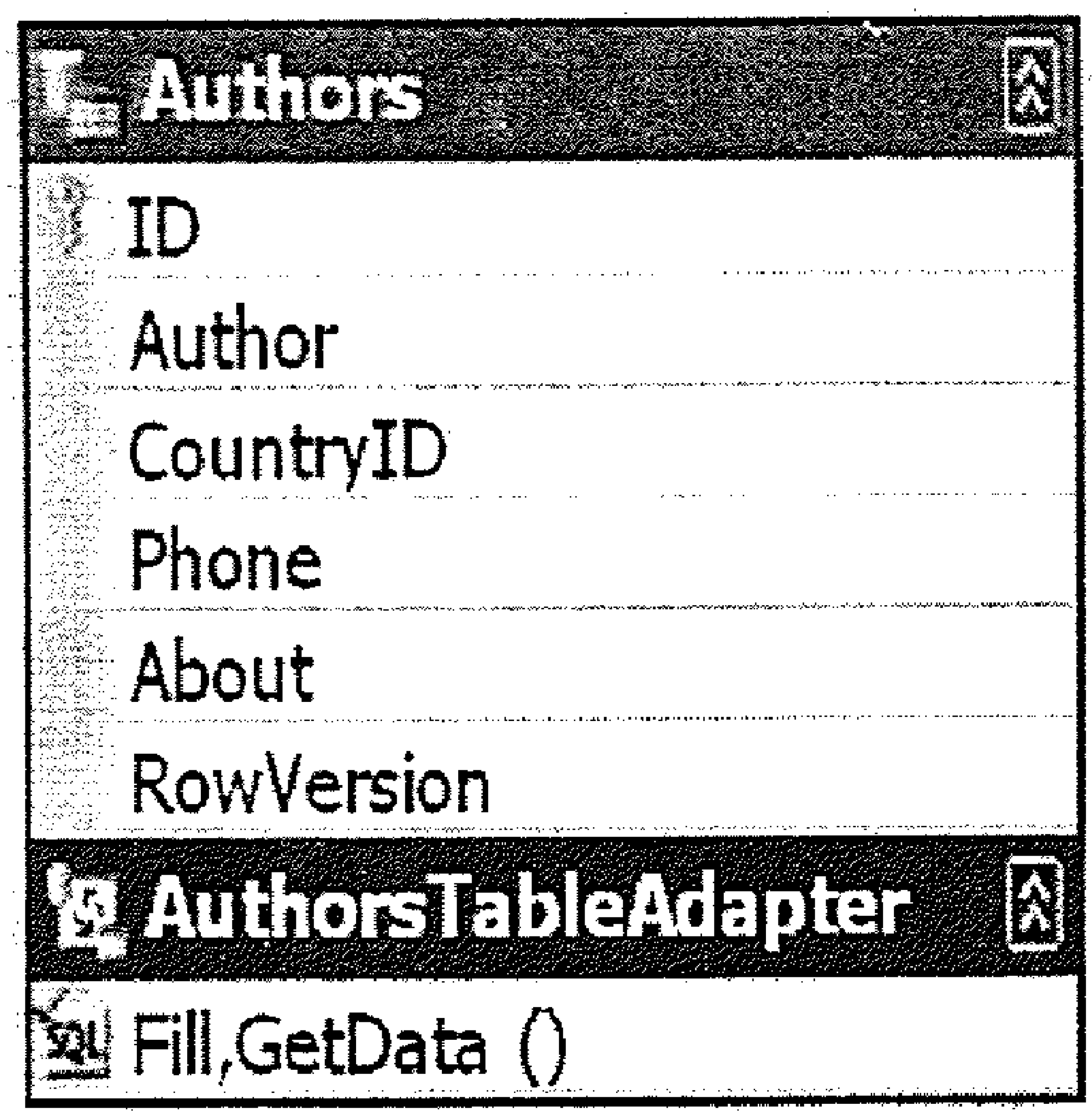

يمكنك أن تختار إنشاء وسيلة لملء جدول موجود في مجموعة البيانات، وسيكون اسمها المبدئي Fill ويمكنك كتابة أي اسم آخر في مربع النص.

كما يمكنك أن تختار إنشاء وسيلة تعيد جدول بيانات DataTable مملوء بالنتائج، لتستخدمه أنت بالطريقة التي تناسبك، وسيكون اسم هذه الوسيلة مبدئيا GetData ويمكنك كتابة أي اسم آخر في مربع النص.

أما الاختيار الأخير، فيجعل موصل الجدول ينشئ الوسائل اللازمة لتحديث قاعدة البيانات.. هذه الوسائل ستحمل الأسماء Update و Insert و Delete.

بعد أن تحدد الاختيارات التي تناسبك، اضغط Next.

- ستظهر نافذة تلخص اختياراتك.. اضغط Finish لإنهاء المعالج السحري وإنشاء موصل الجدول.




ستجد في مصمم المخطط العنصر الموضح في الصورة.. هذا العنصر يمثل مخطط الجدول Authors، وفي الجزء السفلي منه مخطط موصل الجدول الذي سيستخدم للتعامل معه، واسمه الافتراضي AuthorsTableAdapter.
ويمكنك ضغط الشريط الذي يعرض اسم موصل الجدول بزر الفأرة الأيمن، واختيار الأمر Properties لعرض خصائص موصل الجدول في نافذة الخصائص.. وهذه الخصائص هي:

### الاسم Name:

تحدد اسم فئة موصل الجدول.

### المجال Modifier:

تحدد مجال فئة الجدول.. والقيمة الافتراضية هي Public ليمكن استخدام هذه الفئة حتى من خارج المشروع الحالي.

### الفئة الأم Base Class:

تحدد الفئة الأم التي ترثها فئة موصل الجدول.. في الوضع الافتراضي تكون هذه الفئة هي فئة المكون System.ComponentModel.Component.. لكن لا مانع من أن تكتب بدلا منها أية فئة أخرى بشرط أن تكون مشتقة من الفئة Component.. يمكنك مثلا أن ترث فئة موصل البيانات، أو يمكنك أن ترث فئة موصل جدول آخر!

### الاتصال Connection:

تحدد الاتصال بقاعدة البيانات.. ويمكنك اختيار اتصال من القائمة المنسدلة، أو ضغط العنصر الأخير فيها (New Connection) لإنشاء اتصال جديد.

### مجال الاتصال ConnectionModifier:

تحدد مجال كائن الاتصال المعرف في موصل الجدول.. والقيمة الافتراضية هي Friend لجعله مرئيا من أي موضع في المشروع.

**أمر التحديد SelectCommand:**
أمر التحديد المستخدم لإحضار البيانات من قاعدة البيانات.. لاحظ أنك لو استخدمت أمر تحديد يعيد بيانات من أكثر من جدول، فسيعجز موصل الجدول عن إنتاج أوامر التحديث والإدراج والحذف آليا، لهذا يتوجب عليك في هذه الحالة استخدام الخصائص التالية لتعريف هذه الأوامر بنفسك.

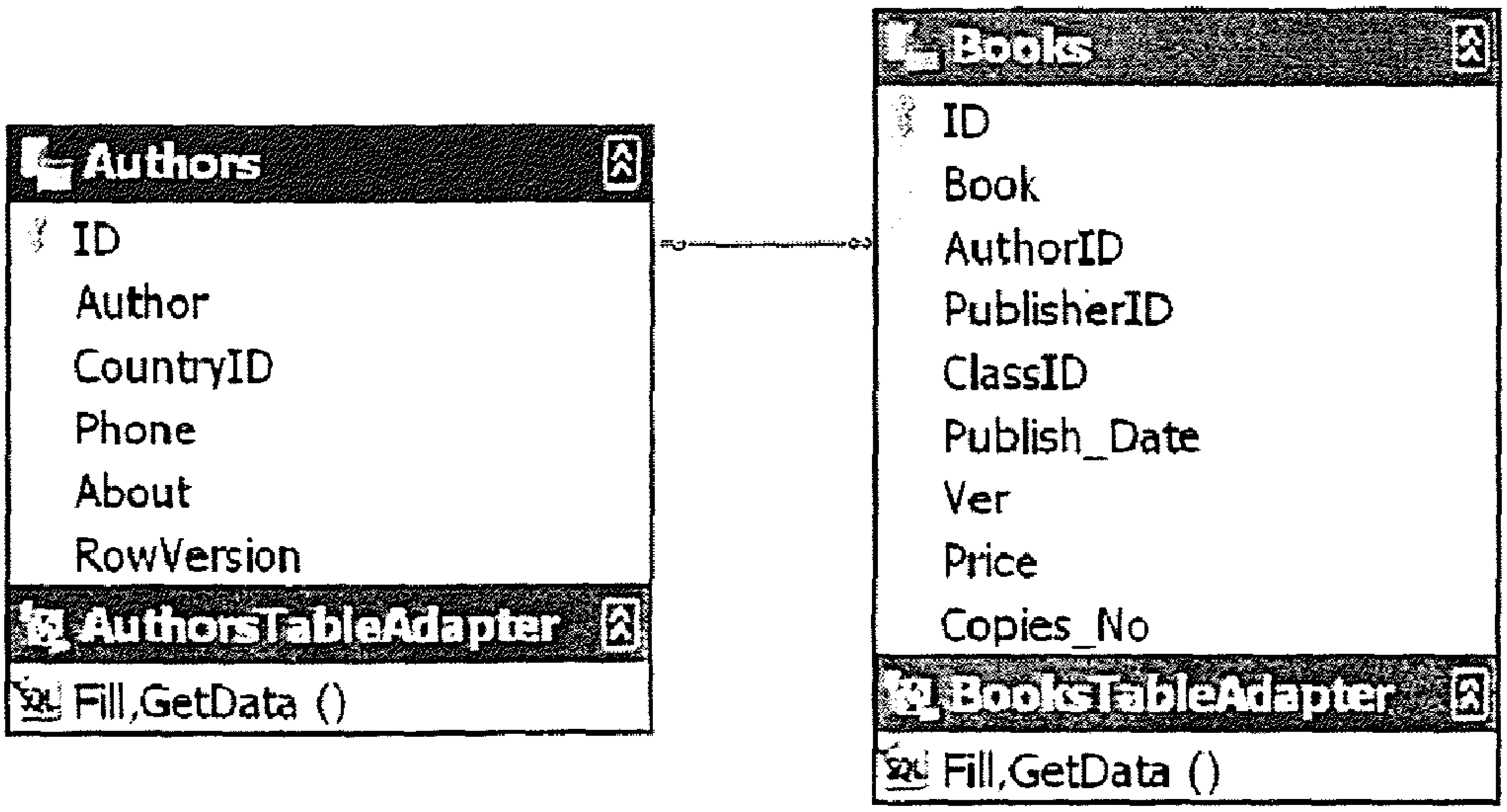
**أمر التحديث UpdateCommand:**
أمر التحديث المستخدم لتحديث سجلات قاعدة البيانات.

**أمر الإدراج InsertCommand:**
أمر الإدراج المستخدم لإدراج السجلات في قاعدة البيانات.

**أمر الحذف DeleteCommand:**
أمر الحذف المستخدم لحذف السجلات من قاعدة البيانات.

**إنتاج وسائل قاعدة البيانات المباشرة GenerateDbDirectMethods:**
إذا جعلت قيمة هذه الخاصية True، فستضاف إلى فئة موصل الجدول الوسائل Update و Insert و Delete لتتيح لك تحديث قاعدة البيانات باستدعائها مباشرة.. أما إذا جعلتها False، فسيكون عليك استخدام كائنات الأوامر بنفسك لإجراء عمليات التحديث والإدراج والحذف، وذلك من خلال الخصائص UpdateCommand، InsertCommand، DeleteCommand.

وبنفس الطريقة يمكنك إضافة موصل جدول الكتب.. لاحظ أن العلاقة بين الجدولين سيتم إنشاؤها تلقائيا بمجرد إضافة جدول الكتب.. هكذا سيكون المخطط:



ولكن، أين يتم إنشاء فئة موصل البيانات؟
يتم إنشاء هذه الفئة في نفس الملف الذي توجد فيه فئة مجموعة البيانات محددة النوع، وهو في مثالنا هذا الملف DsAuthorsBooks.Designer.vb.. لكن فئة موصل الجدول لا توضع داخل فئة مجموعة البيانات، بل توضع خارجها، ويتم تعريف نطاق اسم Namespace خاص بها يكون على الصيغة XTableAdapters، حيث X هو اسم فئة عة البيانات.. هكذا مثلا سيكون الشكل العام لملف كود فئة البيانات وموصلات الجداول في مشروعنا هذا:

```
Partial Public Class DsAuthorsBooks
    Inherits Global.System.Data.DataSet
    ' كود مجموعة البيانات محددة النوع
End Class

Namespace DsAuthorsBooksTableAdapters
    Partial Public Class AuthorsTableAdapter
        Inherits Global.System.ComponentModel.Component
        ' كود موصل جدول المؤلفين
    End Class

    Partial Public Class BooksTableAdapter
        Inherits Global.System.ComponentModel.Component
        ' كود موصل جدول الكتب
    End Class

    Partial Public Class TableAdapterManager
        Inherits Global.System.ComponentModel.Component
        ' كود مدير موصلات الجداول
    End Class
End Namespace
```

وتمتلك فئة موصل الجدول الخصائص التالية:

### الاتصال Connection:

تقرأ أو تغير كائن الاتصال الذي يستخدمه موصل الجدول للاتصال بقاعدة البيانات.. ويعتمد نوع هذه الخاصية على نوع قاعدة البيانات التي تتعامل معها، وفي مثالنا هذا ستكون من النوع SqlConnection.
ويستخدم موصل الجدول إجراءا خاصا Private Sub اسمه InitConnection لضبط خصائص كائن الاتصال.

### الانتقالات Transaction:

تقرأ أو تغير كائن الانتقالات الذي يستخدمه موصل الجدول للتحكم في العمليات التي تتم على قاعدة البيانات.. وفي مثالنا هذا ستكون هذه الخاصية من النوع SqlTransaction.

### محو قبل الملء ClearBeforeFill:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتم محو السجلات من جدول مجموعة البيانات أولا، قبل ملئه بالنتائج الجديدة.. أما إذا جعلتها False، فسيتم تحديث السجلات الموجودة بالقيم الجديدة، وإضافة السجلات الجديدة إلى الجدول.

| ملحوظة: |
| --- |
| موصل الجدول يستخدم موصل البيانات داخليا لملء مجموعة البيانات، لهذا ستجد في فئة موصل الجدول خاصية محمية Protected Property اسمها Adapter، لن تراها من خارج الفئة، لكن تستطيع استخدامها في الفئات التي ترث فئة موصل الجدول، أو في أي كود إضافي تكتبه في موصل الجدول بنفسك.. ويستخدم موصل الجدول إجراء خاصا Private Sub اسمه InitAdapter لوضع القيم في خصائص موصل البيانات.<br>كما يحتوي موصل الجدول على خاصية محمية أخرى اسمها CommandCollection، وهي تعيد مصفوفة تحتوي على كائنات الأوامر التي يستخدمها موصل الجدول، وبهذا يستطيع موصل الجدول التعامل مع أكثر من استعلام كما سنرى لاحقا.<br>ويستخدم موصل الجدول إجراء خاصا Private Sub اسمه InitCommandCollection لوضع كائنات الأوامر في هذه المصفوفة وضبط خصائصها. |

كما تمتلك فئة موصل الجدول الوسائل التالية:

- **ملء Fill:**
تستقبل معاملا من نوع الجدول المراد ملؤه بالبيانات، وتعيد عددا صحيحا يخبرك بعدد السجلات التي أضيفت أو تم تحديثها في هذا الجدول.. وفي موصل جدول المؤلفين، يكون معامل هذه الوسيلة من نوع جدول المؤلفين المعرف في مجموعة البيانات DsAuthorsBooks.AuthorsDataTable، وبالمثل يكون هذا المعامل في موصل بيانات الكتب، من النوع DsAuthorsBooks.BooksDataTable.
لاحظ أنك تستطيع أن ترسل جدولا من جداول مجموعة البيانات إلى هذه الوسيلة، أو ترسل جدول حرا ليس مرتبطا بمجموعة بيانات، المهم أن يكون من النوع الصحيح.

- **قراءة البيانات GetData:**
لا تستقبل أية معاملات، لكنها تعيد جدولا جديدا مملوءا بالبيانات.. هذا الجدول يكون من النوع AuthorsDataTable في موصل جدول المؤلفين، ومن النوع BooksDataTable في موصل بيانات الكتب.

- **تحديث Update:**
تحفظ التغييرات في قاعدة البيانات.. لاحظ أن كل ما تفعله هذه الوسيلة، هو استدعاء الوسيلة Update الخاصة بموصل البيانات الداخلي.. ولهذه الوسيلة الصيغ التالية:
١- الصيغة الأولى تستقبل كائن الجدول المراد حفظ تغييراته.
٢- الصيغة الثانية تستقبل كائن مجموعة البيانات، حيث يقوم موصل الجدول بقراءة التغييرات من الجدول الخاص به في مجموعة البيانات، دون غيره من الجداول.. مثلا: تستخدم هذه الصيغة الكود التالي في موصل جدول المؤلفين:

```
Return Me.Adapter.Update(dataSet, "Authors")
```

٣- الصيغة الثالثة تستقبل كائن صف البيانات DataRow الذي تريد حفظ تغييراته في قاعدة البيانات.
٤- الصيغة الرابعة تستقبل مصفوفة تحتوي على صفوف البيانات التي تريد حفظ تغييراتها في قاعدة البيانات.

٥- الصيغة الخامسة تستقبل قيم الصف المراد حفظه في قاعدة البيانات.. ولهذه الصيغة عدة معاملات، كل منها يستقبل قيمة أحد الأعمدة الموجودة في الصف.. مثلا، ستحتوي هذه الوسيلة في موصل بيانات المؤلفين على هذه المعاملات بالترتيب: Author، CountryID، Phone، About، Original_ID، Original_RowVersion.

٦- الصيغة السادسة تزيد على الصيغة السابقة بمعامل إضافي يستقبل المفتاح الأساسي للجدول (الحقل ID في مثالنا هذا).

وتعيد هذه الوسيلة عددا صحيحا يخبرك بعدد السجلات التي تم تحديثها في قاعدة البيانات، فإذا كان الناتج صفرا، فهذا معناه حدوث مشكلة تطابق Concurrency Violation.

### إدراج Insert:

تدرج صفا جديدا في قاعدة البيانات.. ولهذه الوسيلة عددا من المعاملات بعدد أعمدة الجدول، لاستقبال قيم الصف المراد إضافته.

وتعيد هذه الوسيلة عددا صحيحا يخبرك بعدد السجلات التي أضيفت إلى قاعدة البيانات، فإذا كان الناتج صفرا، فهذا معناه أن قاعدة البيانات رفضت إدراج الصف بسبب وجود مشكلة في قيمة إحدى خاناته.

### حذف Delete:

تحذف سجلا من قاعدة البيانات.. وتميز هذه الوسيلة السجل باستقبال مفتاحه الأساسي Original_ID وإصداره Original_RowVersion كمعاملين..

لاحظ أننا لا نستخدم إصدار السجل في جدول الكتب، لهذا تمتلك هذه الوسيلة في موصل جدول الكتب معاملات بعدد حقول الجدول، للبحث عن السجل الأصلي في قاعدة البيانات بدلالة كل قيمه.

وتعيد هذه الوسيلة عددا صحيحا يخبرك بعدد السجلات التي تم حذفها من قاعدة البيانات، فإذا كان الناتج صفرا، فهذا معناه حدوث مشكلة تطابق Concurrency Violation.

### إضافة استعلامات جديدة إلى موصل الجدول:

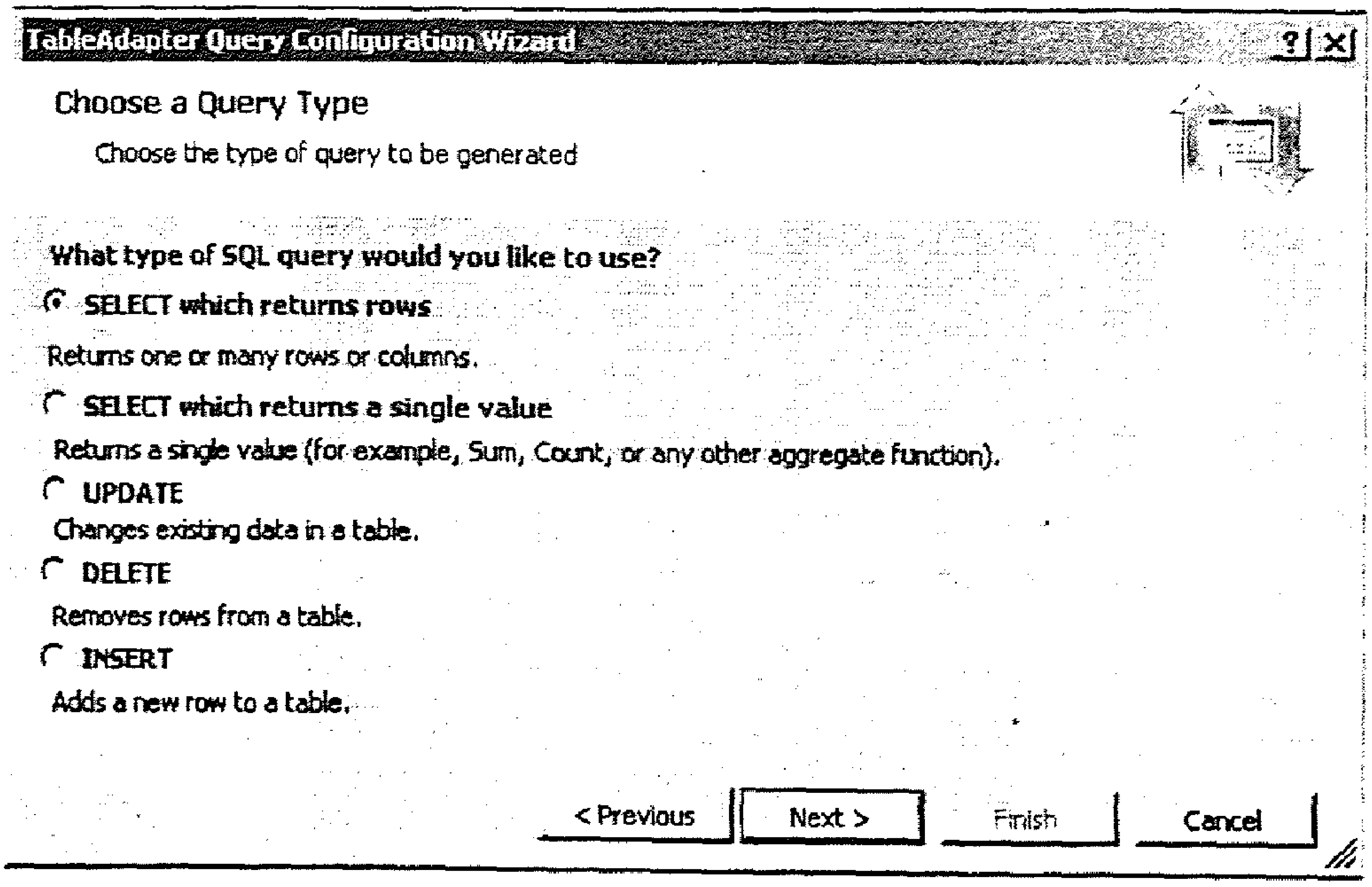
إلى الآن، لا يبدو أن موصل الجدول يقدم شيئا جديدا يميزه عن موصل البيانات العادي.. فالحقيقة أن مزية موصل الجدول الرئيسية، هي قدرتك على إضافة أي عدد تريده من الاستعلامات إليه، ما دامت تلتزم بأحد الشرطين التاليين:

١- أن تعيد سجلات لها نفس تركيب الجدول الذي يتعامل معه موصل الجدول.. ليس من المنطقي مثلا أن تضيف إلى موصل جدول المؤلفين، استعلاما يعيد سجلات الكتب.

٢- أن تعيد قيمة منفردة Scalar Value.. يمكنك مثلا أن تضيف إلى موصل جدول المؤلفين استعلاما يعيد عدد المؤلفين، أو عدد كتب أحد المؤلفين.

ولإضافة استعلام جديد إلى موصل الجدول، اضغط اسم الموصل في نافذة المصمم بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Add Query.. سيؤدي هذا إلى بدء المعالج السحري لتهيئة استعلام موصل الجدول TableAdapter Query Configuration Wizard.. دعنا نستخدم هذا المعالج لإضافة استعلام إلى موصل جدول الكتب، يعيد كتب المؤلف الذي نريده:

- النافذة الأولى تسألك عن نوع الاستعلام الذي تريده (جملة SQL أم إجراء مخزن).. اختر Use SQL Statement واضغط Next.
- النافذة التالية تسألك عن نوع الاستعلام الذي تريده:




يمكنك الاختيار من بين الأنواع التالية:

أ. جملة استعلام تعيد صفوفا:

SELECT statement witch returns rows.

ب. جملة استعلام تعيد قيمة منفردة:

SELECT statement witch returns a single value.

ت. تحديث UPDATE.

ث. حذف DELETE.

ج. إدراج INSERT.

اختر أول اختيار، واضغط Next.

- في النافذة التالية اكتب جملة الاستعلام التالية:

```
SELECT    Books.*
FROM Books INNER JOIN Authors
ON Books.AuthorID = Authors.ID
AND Author = @Author
```

واضغط Next.

- النافذة التالية تتيح لك اختيار الوسائل التي ستضاف إلى موصل جدول الكتب لتنفيذ هذا الاستعلام.. ستجد وسيلتين هما:

أ. FillBy، وعليك تعديل اسمها إلى FillByAuthor، وهي تستقبل اسم المؤلف، وتملأ جدول الكتب في مجموعة البيانات بكتب هذا المؤلف.

ب. GetDataBy، وعليك تعديل اسمها إلى GetDataByAuthor، وهي تستقبل اسم المؤلف، وتعيد جدول كتب يحتوي على كتب هذا المؤلف.

- اضغط Next لعرض نافذة الملخص، ثم اضغط Finish لإنهاء المعالج السحري.

سيؤدي هذا إلى ظهور اسمي الوسيلتين الجديدتين في مخطط موصل جدول الكتب، كما هو موضح في الصورة المجاورة.. هذا معناه أن تعريف هاتين الوسيلتين قد أضيف إلى فئة موصل الجدول، وسيكون لكل منهما معامل نصي يستقبل اسم المؤلف.. وعموما، يقوم موصل جدول الكتب بتعريف المعاملات المناسبة لنوع الحقل الذي تستعلم عنه في قاعدة البيانات.

لاحظ أنك لو استخدمت استعلاما يعيد نتائج غير مرغوبة، فسيعرض لك موصل الجدول رسالة تحذرك من أن نتيجة الاستعلام لا تناسب مخطط الجدول.. ولو أردت تصحيح الاستعلام فاضغط بزر الفأرة الأيمن، فوق الصف الذي يعرض

اسمي الوسيلتين الجديتين في مخطط موصل الجدول، ومن القائمة الموضعية اضغط الأمر Configure.. سيعرض هذا النافذة التي أدخلت فيها الاستعلام، حيث يمكنك تصحيحه كما تريد وضغط الزر Finish.

ولحذف الاستعلام، حدده في مخطط موصل الجدول، واضغط Delete.

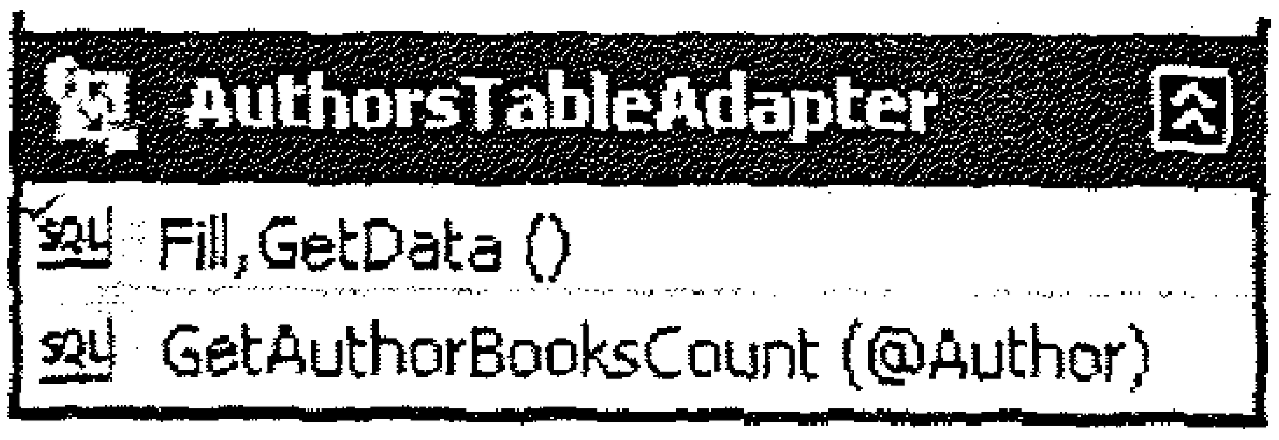
دعنا أيضا ننشئ استعلاما في موصل جدول الكتب يعيد لنا عدد كتب مؤلف معين.. دعنا نجرب طريقة أخرى هذه المرة.. من صندوق الأدوات اسحب العنصر Query وأسقطه فوق موصل جدول المؤلفين.. سيؤدي هذا إلى إطلاق المعالج السحري، حيث يمكنك اتباع نفس الخطوات السابقة، لكن مع اختيار:

SELECT Statement that returns a single value

وفي نافذة الاستعلام اكتب:

**SELECT COUNT(BOOK) FROM Authors, Books**
**WHERE AuthorID = Authors.ID AND Author = @Author**

واضغط Next.. ستظهر لك نافذة تتيح لك تسمية الدالة التي تنفذ هذا الاستعلام.. سيكون لهذه الدالة الاسم الافتراضي ScalarQuery.. غير هذا الاسم إلى GetAuthorBooksCount واضغط Finish.. سيظهر اسم الدالة الجديدة في مخطط موصل الجدول كما في الصورة، كما ستضاف هذه الوسيلة إلى فئة موصل الجدول، حيث ستستقبل نصا يمثل اسم المؤلف، وتعيد عددا صحيحا يمثل عدد كتبه.



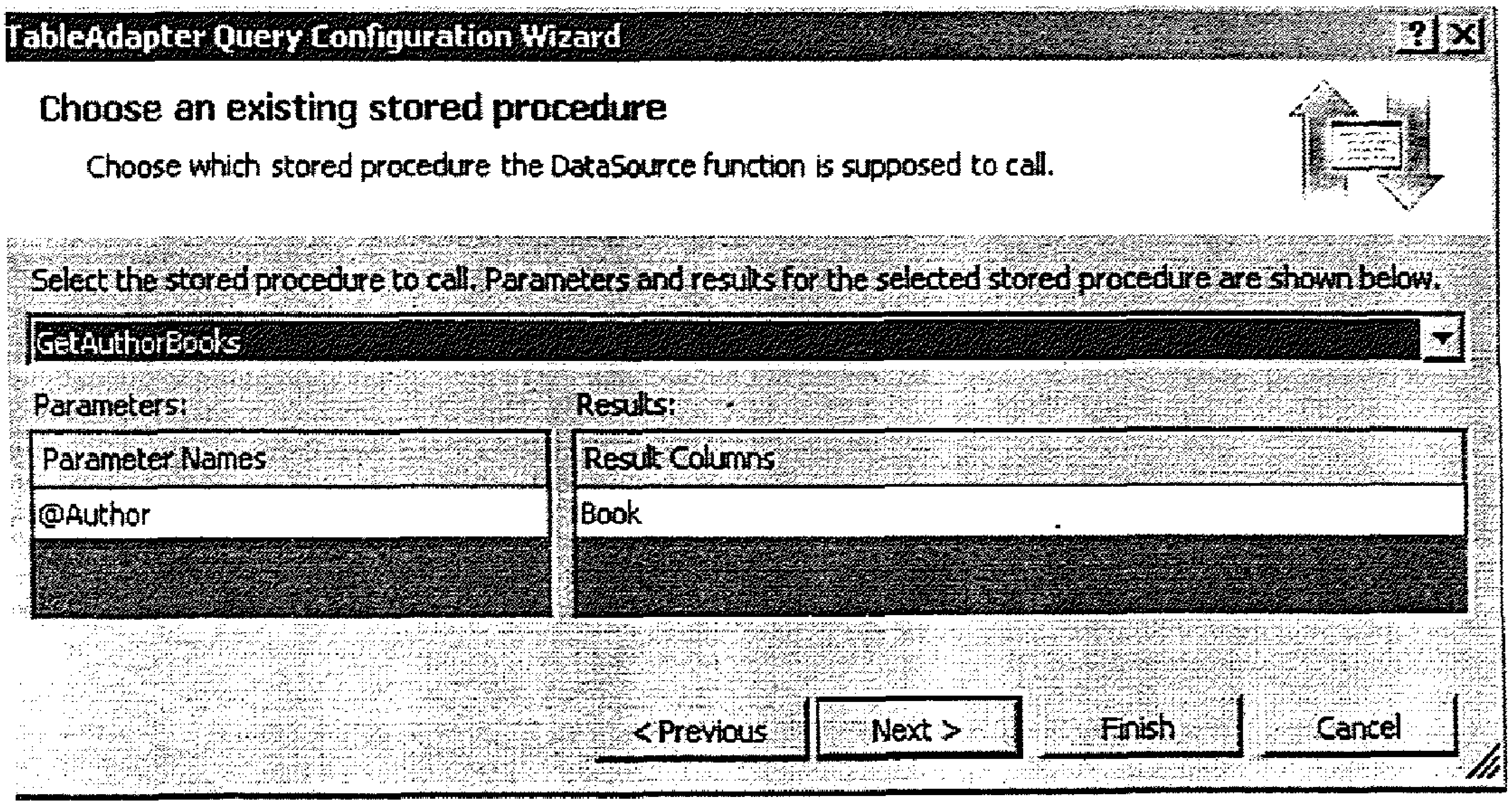

| ملحوظة1: |
| --- |
| عند إنشاء الاستعلام عن حقل يمكن تركه فارغا (مثل الحقل Phone في جدول الكتب)، يقوم موصل البيانات بتعريف معامل الوسيلة FillBy بحيث يكون قابلا للانعدام Nullable (مثلا: سيكون معامل الوسيلة FillByPhone من نوع النص المنعدم ?String).. هذا يتيح لك إرسال القيمة Nothing إلى هذه الوسيلة لتعيد إليك السجلات التي ما زال فيها هذا الحقل فارغا. |

**ملحوظة٢:**

إذا أردت إضافة وسيلة لتنفيذ إجراء مخزن، فاتبع نفس الخطوات المألوفة لإضافة استعلام، لكن هذه المرة اختر نوع الاستعلام:

Existing Stored Procedure

واضغط Next.. ستظهر نافذة تعرض قائمة منسدلة بها أسماء الإجراءات المخزنة في قاعدة البيانات.. اختر الإجراء GetAuthorBooks.. سيعرض النصف السفلي من النافذة بيانات هذا الإجراء المخزن: على اليسار ستظهر معاملات الإجراء، وعلى اليمين ستظهر الأعمدة الناتجة عن تنفيذه، كما هو موضح في الصورة:



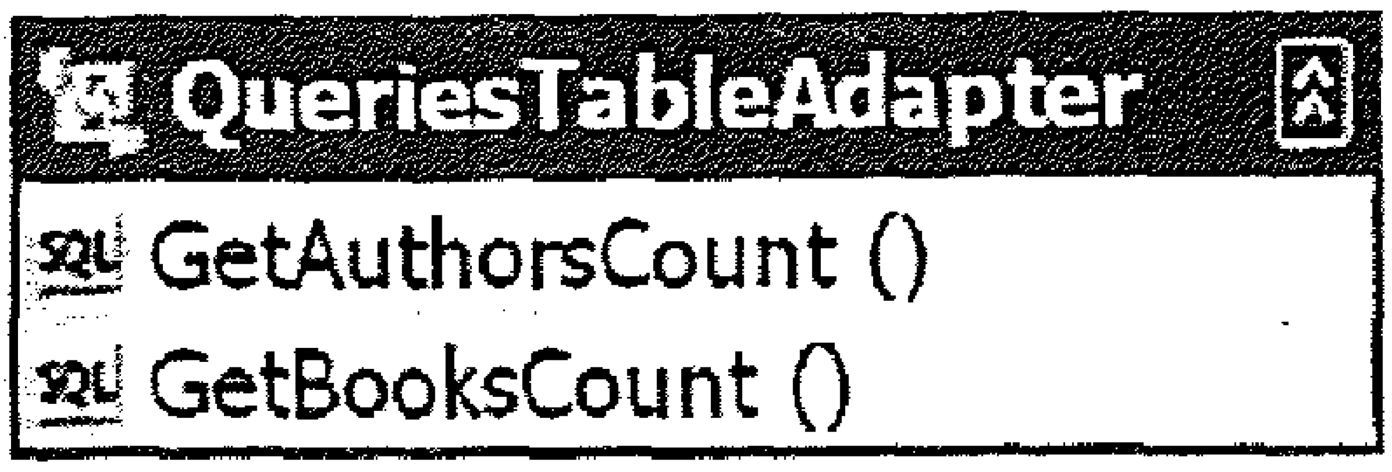

اضغط Next للانتقال إلى النافذة التالية، وهي تسألك عن القيمة العائدة من الوسيلة التي ستنفذ الإجراء المخزن.. هذه القيمة قد تكون:

- قيمة جدولية Tabular Value، حيث تعيد الوسيلة كائن جدول يحتوي على الصفوف الناتجة.
- قيمة منفردة Single Value، حيث تعيد الوسيلة قيمة أول خانة في أول عمود في النتيجة.
- ولا قيمة No Value، حيث ستكون الوسيلة بدون قيمة عائدة، وهذا مناسب للإجراءات المخزنة التي لا تعيد ناتجا.

اختر ما يناسبك واضغط Next.. باقي الخطوات لا جديد فيها.

### إنشاء استعلامات عامة Global Queries:

يمكنك إنشاء موصل جدول لتنفيذ استعلامات عامة، كحساب دالة تجميع، أو تنفيذ استعلامات الحذف والإدراج دون أن يكون مرتبطا بجدول معين في مجموعة البيانات.. لفعل هذا، اضغط بزر الفأرة الأيمن في أي منطقة خالية من مصمم مجموعة البيانات، ومن القائمة الفرعية Add اضغط الأمر Query.. أو اسحب العنصر Query من صندوق الأدوات وألقه على أي منطقة فارغة من مصمم مجموعة البيانات.. سيؤدي هذا إلى إطلاق المعالج السحري لتهيئة الاستعلام، لكنه سيبدأ هذه المرة بنافذة اختيار الاتصال بقاعدة البيانات، ثم يستمر بنفس الخطوات السابقة، لكنك لن تستطيع إنشاء استعلام يعيد سجلات بهذا المعالج.. يمكنك فقط إنشاء استعلامات تعيد قيما منفردة، أو استعلامات التحديث والحذف والإدراج.. وبعد أن تنهي المعالج، ستجد موصل جدول جديد قد أضيف إلى مجموعة البيانات، وسيكون اسمه QueriesTableAdapter وهو اسم لا يمكنك تغييره.. وأية استعلامات عامة أخرى ستنشئها ستضاف إلى هذا الموصل.. وقد أضفنا إليه في مشروعنا هذا الدالة GetAuthorsCount التي تعيد عدد المؤلفين، والدالة GetBooksCount التي تعيد عدد الكتب.

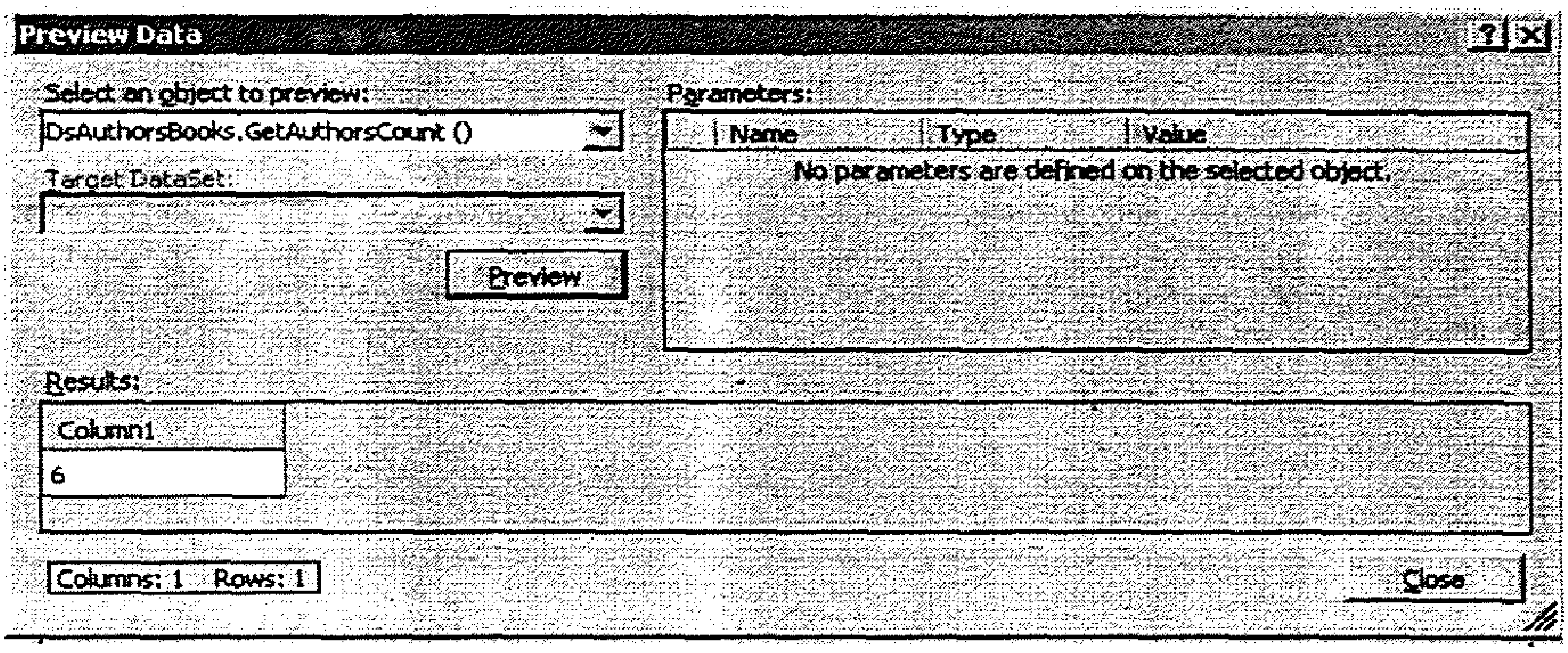


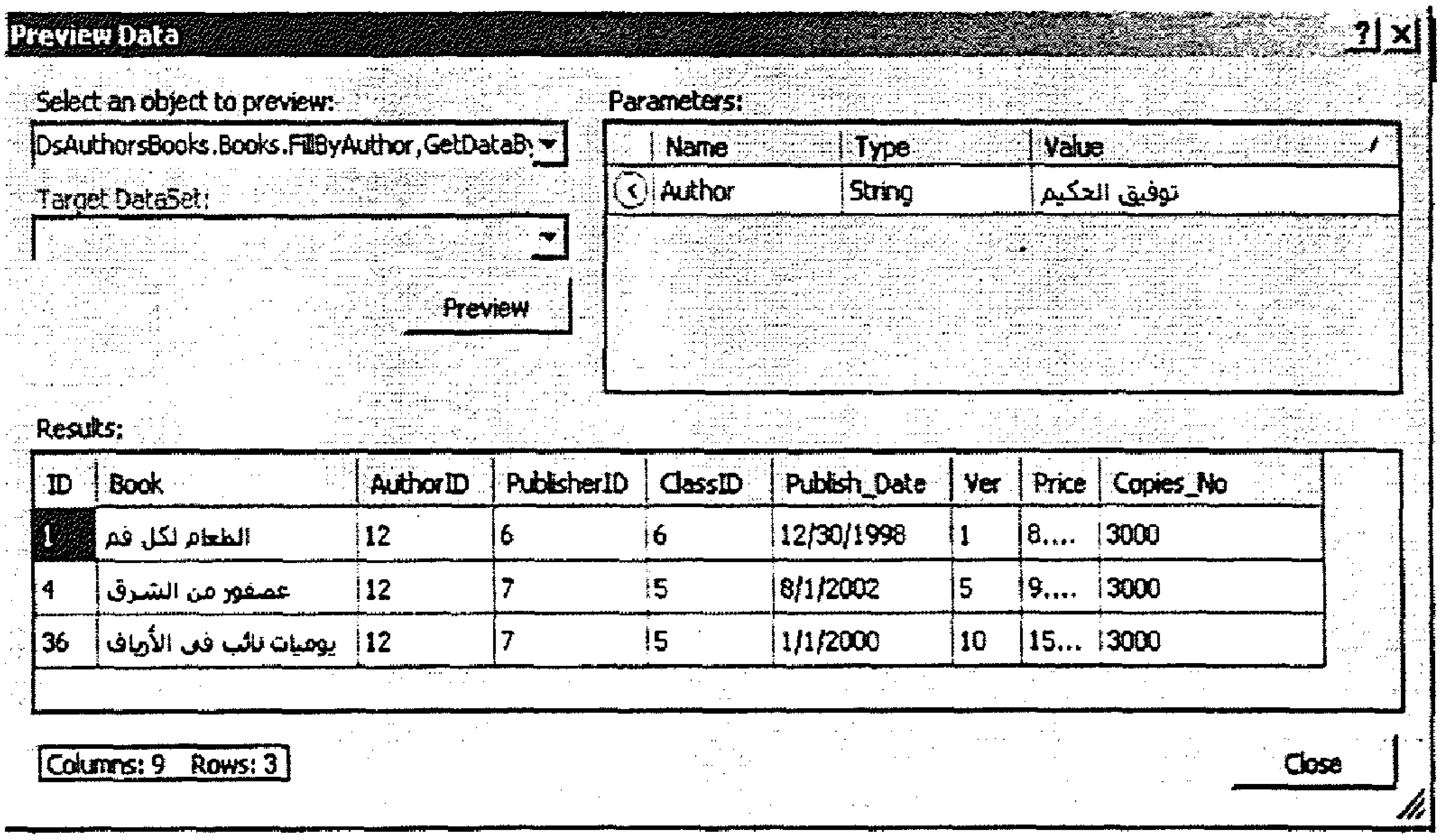

تذكر مرة أخرى، أن موصل جدول الاستعلامات لا يتعامل مع استعلامات أو إجراءات مخزنة تعيد سجلات.. هو فقط يتعامل مع استعلامات أو إجراءات مخزنة تعيد قيما مفردة، وإذا اخترت إجراءا مخزنا يعيد سجلات، فستعيد هذه الوسيلة قيمة أول خانة في أول صف في النتيجة!

**معاينة بيانات مجموعة البيانات:**

يمكنك معاينة نتيجة أي دالة في موصل الجدول، بالضغط بزر الفأرة الأيمن في أي موضع خال في مصمم مجموعة البيانات، أو فوق تصميم أحد الجداول أو أحد موصلات الجداول، وضغط الأمر Preview Data من القائمة الموضعية.. وستجد نفس الأمر في القائمة الرئيسية Data.

ستظهر نافذة عرض النتائج، حيث يمكنك إسدال القائمة العلوية، واختيار موصل الجدول الذي تريده، والدالة التي تريد تنفيذها منه، ثم وضع المعاملات في الجدول الموجود على يمين النافذة إن كانت الدالة تحتاج معاملات، ثم ضغط الزر Preview لرؤية النتيجة.



## فئة مدير موصلات الجداول TableAdapterManager

ظهرت هذه الفئة في دوت نت ٢٠٠٨، ويتم إنتاجها آليا عند إنشاء موصلات الجداول التي تربطها علاقات، لتسمح لك بإجراء التحديث المتراكب Hierarchical Update، وفيه يتم تحديث الجداول المترابطة معا، مع قدرتك على تحديد الترتيب الصحيح لإجراء عمليات التحديث، لمراعاة القيود المفروضة على الجداول.
لاحظ أنك تستطيع منع إنتاج هذه الفئة.. لفعل هذا اضغط بزر الفارة الأيمن في أي موضع فارغ من مخطط مجموعة البيانات، ومن القائمة الموضعية اضغط الأمر Properties، وفي نافذة الخصائص غير قيمة الخاصية Hierarchical Update إلى False.

وتمتلك هذه الفئة الخصائص التالية:

### الاتصال Connection:

تقرأ أو تغير كائن الاتصال المستخدم في عملية التحديث.

### مدير موصل الجدول... XTableAdapter:

الحرف X الذي وضعناه في بداية اسم هذه الخاصية هو بديل عن اسم أحد الجداول.. هذا معناه أن مدير موصلات الجداول يمتلك خاصية لكل موصل جدول تم تعريفه في مجموعة البيانات.. وفي مشروعنا هذا ستحتوي فئة المدير على الخاصيتين AuthorsTableAdapter و BooksTableAdapter.
وتكون لهذه الخصائص القيمة Nothing إلى أن تضع في كل منها موصل الجدول الذي تريد أن يتحكم فيه مدير الموصلات.

### عدد نسخ موصلات الجداول TableAdapterInstanceCount:

تعيد عدد نسخ موصلات الجداول التي لها قيمة غير Nothing.

### نسخ مجموعة البيانات احتياطيا قبل التحديث :BackUpDataSetBeforeUpdate

إذا جعلت قيمة هذه الخاصية True، فسيتم حفظ نسخة احتياطية من مجموعة البيانات قبل إجراء عملية التحديث.. يحدث هذا بتعريف مجموعة بيانات داخل إجراء التحديث، وحفظ سجلات مجموعة البيانات الأصلية فيها.. هذا مفيد إذا حدث خطأ أثناء عملية التحديث، ففي هذه الحالة سيتم التراجع Rollback عن كل العمليات التي أجريت على قاعدة البيانات، وستستعيد مجموعة البيانات الأصلية حالتها السابقة قبل إجراء عملية التحديث، وذلك باستعادتها من مجموعة البيانات الاحتياطية.. لاحظ أن أخذ نسخة احتياطية من مجموعة بيانات ضخمة سيكون عبئا على الذاكرة وسيستهلك وقتا لتنفيذه، لهذا فالقيمة الافتراضية لهذه الخاصية هي False.

### ترتيب التحديث :UpdateOrder

تحدد ترتيب تنفيذ أوامر التحديث والإدراج والحذف عند إجراء عملية التحديث، وهي تأخذ إحدى قيمتي المرقم UpdateOrderOption التاليتين:

| | |
|---|---|
| InsertUpdateDelete | تنفيذ أوامر الإدراج ثم التحديث ثم الحذف.. هذه القيمة الافتراضية. |
| UpdateInsertDelete | تنفيذ أوامر التحديث ثم الإدراج ثم الحذف. |

وتمتلك هذه الفئة الوسيلة الوحيدة التالية:

### تحديث الكل :UpdateAll

أرسل إلى هذه الوسيلة مجموعة البيانات محددة النوع التي تريد نقل التغييرات منها إلى قاعدة البيانات.. في مشروعنا ستكون مجموعة البيانات من النوع DsAuthorsBooks.. ويتم إجراء عمليات التحديث بالترتيب الموضح في الخاصية UpdateOrder، وإذا حدث خطأ في أي مرحلة من مراحل التحديث، يتم التراجع Rollback عن تنفيذ جميع عمليات التحديث، أي أن قاعدة البيانات لا يحدث بها أي تغيير، وتظل كما كانت قبل استدعاء هذه الوسيلة.

**إضافة أكواد خاصة بك إلى مجموعة البيانات والجداول وموصلات الجداول:**

يمكنك أن تضيف بعض الوسائل إلى فئة مجموعة البيانات، أو فئة الجدول، أو فئة موصل الجدول.. كما يمكنك كتابة إجراءات تستجيب لبعض أحداث فئة الجدول، سواء الأحداث المعرفة داخل فئة الجدول، أو تلك الموروثة من الفئة DataTable، والتي سنتعرف عليها في الفصل التالي.

لكن المشكلة أنك لو كتبت أي كود في الملف X.Designer.vb الذي فيه تعريف هذه الفئات (حيث X هو اسم مجموعة البيانات)، فسيكون هذا الكود عرضة للضياع عند قيامك بأي تعديلات في مصمم مجموعة البيانات، لأن هذه التعديلات ستعيد إنتاج ملف الكود من جديد، وستتخلص من أي كود خاص بك!

لحل هذه المشكلة، تم تعريف الفئات في هذا الملف باعتبارها جزئية Partial، ليمكنك إضافة الكود إليها في ملف آخر.. لفعل هذا، اضغط بزر الفأرة الأيمن فوق الجدول أو موصل الجدول، ومن القائمة الموضعية اختر الأمر View Code، لفتح تعريف جزئي مستقل لفئة الجدول أو فئة موصل الجدول.. هذا التعريف سيضاف في ملف جديد اسمه X.vb (في مثالنا هذا سيكون اسمه DsAuthorsBooks.vb).. وستجد هذا الملف ضمن الملفات الفرعية لمخطط مجموعة البيانات DsAuthorsBooks.xsd.

وعندما تفتح هذا الملف في محرر الكود، يمكنك اختيار الفئة من القائمة العلوية اليسرى، واختيار الحدث الذي تريد إضافته إليها من القائمة العلوية اليمنى كما هو مألوف.. كما يمكنك أن تضيف أية دالة تريدها إلى أية فئة، سواء كانت خاصة Private أو عامة Public، مع قدرتك على استخدام كل العناصر المعرفة على مستوى الفئة في كتابة كود هذه الدالة، سواء كانت هذه العناصر محمية Protected أو خاصة Private.

كما يقدم لك مصمم مجموعة البيانات الكثير من التسهيلات:

- فالنقر مرتين في أية منطقة خالية، يفتح فئة مجموعة البيانات.
- والنقر مرتين على عنوان الجدول يفتح فئة الجدول، ويضيف إليها مستجيبا للحدث XRowChanging، حيث X هو اسم الجدول.. وقد استخدمنا الحدث AuthorsRowDeleting في المشروع TableAdapter لعرض رسالة تأكيد قبل حذف أي صف من الجدول.
- والنقر مرتين على أي صف في الجدول، يفتح فئة الجدول، ويضيف إليها تعريفا للحدث ColumnChanging، وشرطا يتأكد أن العمود الذي تغير هو العمود الذي نقرته بالفأرة.. وقد استخدمنا هذا الحدث في المشروع TableAdapter لمنع المستخدم من ترك اسم المؤلف فارغا.
- والنقر مرتين على عنوان موصل الجدول يفتح كود فئته.

**استخدام موصل الجدول في الكود:**

لاستخدام موصل الجدول يجب أن تعرف نسخة منه.. مثال:

```
Dim TaAuhtors As New _
    DsAuthorsBooksTableAdapters.AuthorsTableAdapter
MsgBox(TaAuhtors.GetAuthorBooksCount("توفيق الحكيم"))
```

ولاستخدام مدير الموصلات في الكود، يجب أن نعرف نسخة منه، وتضع نسخة من كل موصل جدول في الخاصية المناظرة له في مدير الموصلات، كالتالي:

```
Dim TaBooks As New _
    DsAuthorsBooksTableAdapters.BooksTableAdapter
Dim TaM As New _
    DsAuthorsBooksTableAdapters.TableAdapterManager
TaM.AuthorsTableAdapter = TaAuhtors
TaM.BooksTableAdapter = TaBooks
```

لكن الأسهل هو أن نتعامل مع هذه الكائنات بشكل مرئي، حيث تقدم لك دوت نت هذه التسهيلات:

١- عند سحب مجموعة البيانات من صندوق الأدوات وإسقاطها على النموذج، ستظهر نافذة تسألك إن كنت تريد إضافة مجموعة بيانات عادية أم محددة النوع، حيث تستطيع اختيار النوع X.DsAuthorsBooks من القائمة المنسدلة، حيث X هو اسم المشروع (وهو TableAdapter في حالتنا هذه).

٢- يمكنك التعامل مع عناصر مجموعة البيانات محددة النوع من صندوق الأدوات مباشرة، فهي يظهر في صندوق الأدوات تحت شريط جديد اسمه X Components، حيث X هو اسم المشروع.. وإذا لم تجد هذا الشريط، فأغلق صندوق الأدوات ثم أعد فتحه ليتم إنعاشه.. وستجد تحت شريط المشروع TableAdapter العناصر التالية:

أ. مجموعة البيانات محددة النوع DsAuthorsBooks.

ب. موصل جدول المؤلفين AuthorsTableAdapter.

ج. موصل جدول الكتب BooksTableAdapter.

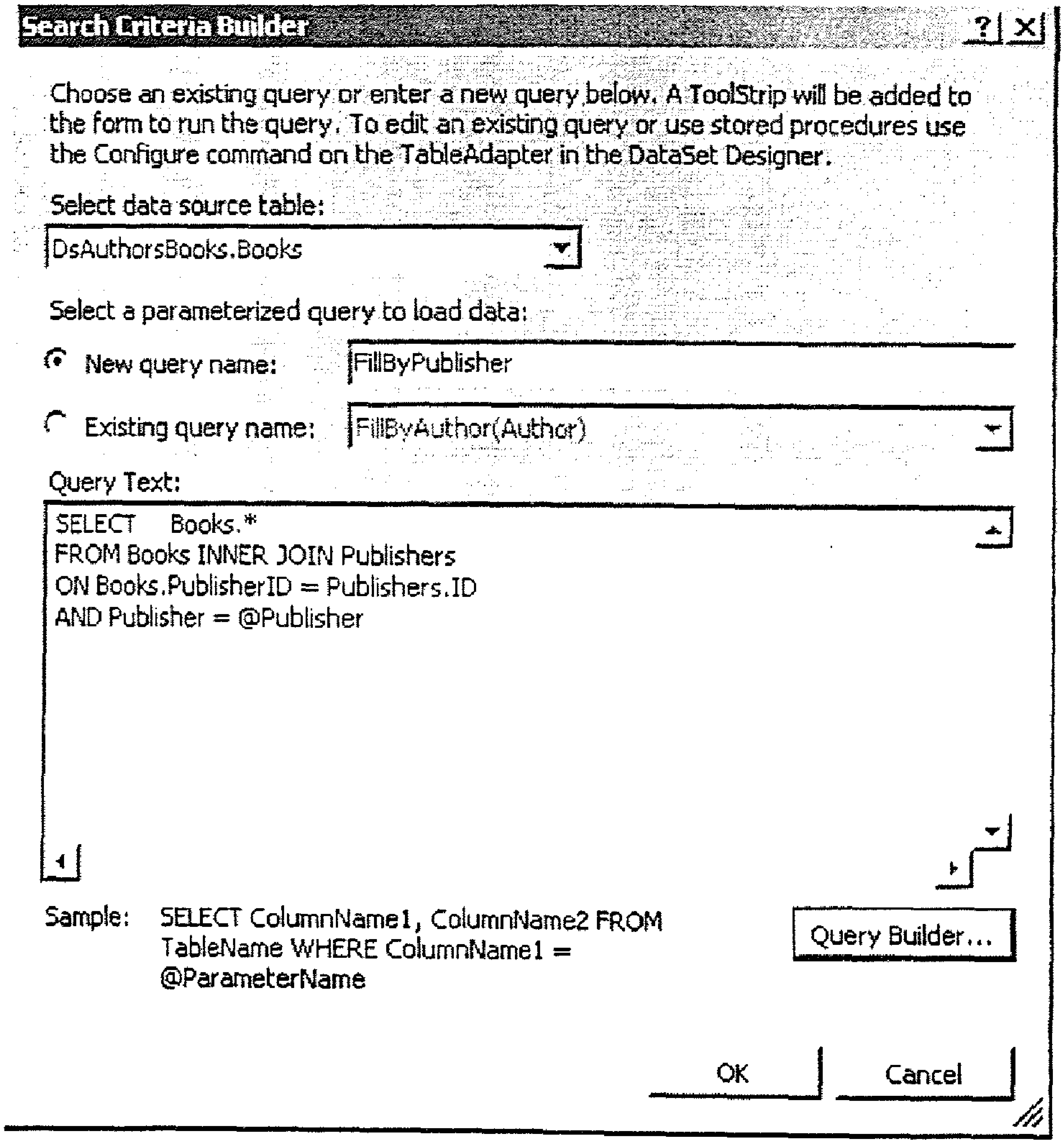
د. موصل جدول الاستعلامات QueriesTableAdapter.

هـ. مدير موصلات الجداول TableAdapterManger.

هذا يتيح لك إضافة أي من هذه العناصر إلى النموذج، حيث ستظهر في صينية المكونات.. أضف نسخة من كل عنصر من هذه العناصر، وامنحها الأسماء التالية على الترتيب: Ds، TaAuthors، TaBooks، TaQueries، TaM.

٣- اضغط بزر الفأرة الأيمن، مدير الموصلات TaM في صينية المكونات، ومن القائمة الموضعية اضغط الأمر Properties لعرض خصائصه في نافذة الخصائص.. من القائمة المنسدلة للخاصية AuthorsTableAdapter اختر TaAuthors، ومن القائمة المنسدلة للخاصية BoosTableAdapter اختر TaBooks.. ويمكنك تغيير قيمة باقي الخصائص كما يناسبك.

٤- اضغط بزر الفأرة الأيمن، أي موصل جدول في صينية المكونات، ومن القائمة الموضعية اضغط الأمر Add Query لإضافة استعلام جديد إلى موصل الجدول.. في هذه الحالة ستظهر نافذة مختصرة، تتيح لك تعريف الوسيلة FillBy فقط ولن يتم تعريف الوسيلة GetDataBy، كما هو موضح في الصورة:



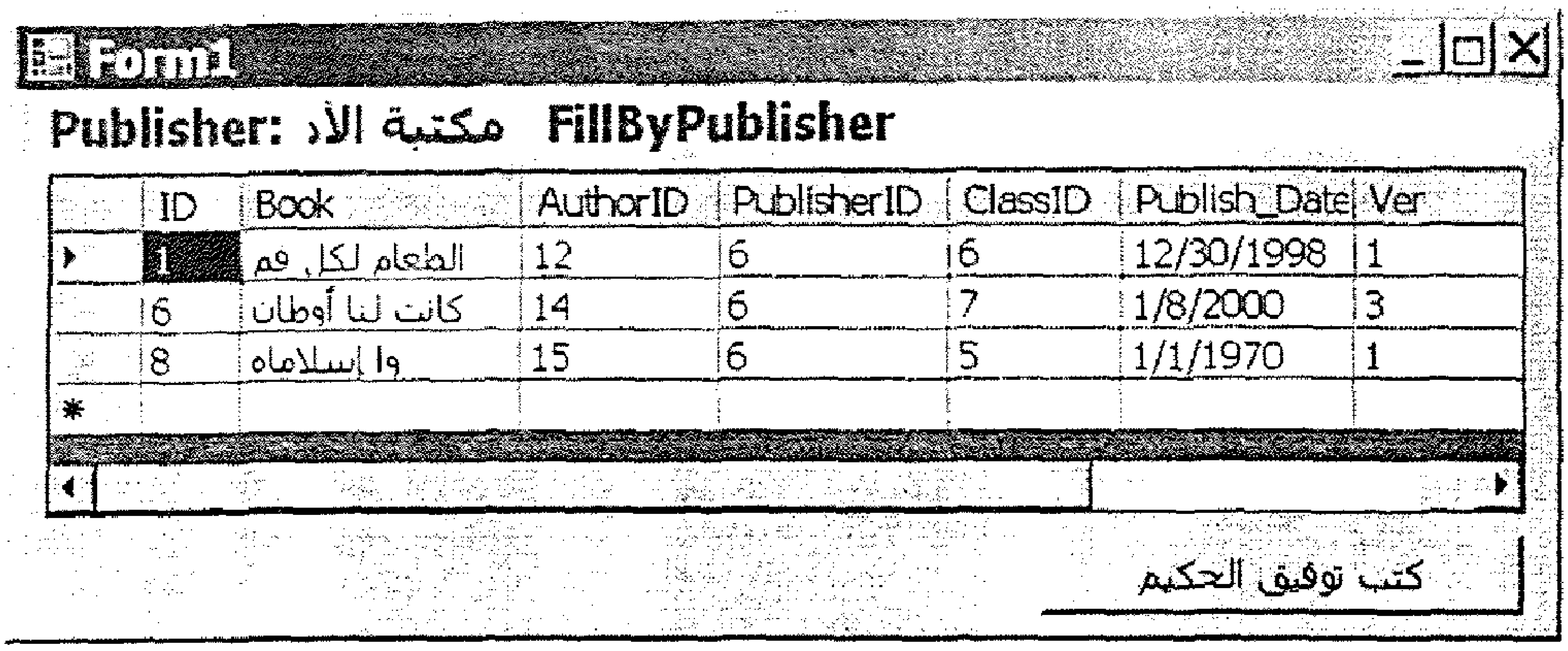
ويمكنك أن تغير موصل الجدول الذي تتعامل معه من القائمة المنسدلة، ويمكنك أن تغير اسم وسيلة الملء بتحريرها في مربع النص (وسنستخدم هنا الاسم FillByPublisher)، كما يمكنك أن تكتب الاستعلام في مربع النص السفلي.. وإذا أردت تعديل استعلام موجود سابقا، فاضغط الاختيار Existing Query Name واختر اسم الاستعلام من القائمة المنسدلة لعرضه في مربع النص السفلي.. في حالتنا هذه، سنستخدم استعلاما جديدا للحصول على الكتب التي نشرها ناشر معين.

اضغط Ok لإغلاق هذه النافذة.. سيؤدي هذا إضافة الوسيلة FillByPublisher إلى موصل بيانات الكتب.

وإذا أردت إضافة الوسيلة GetDataByPublisher، فاتبع الخطوات التالية:

- ضغط موصل الجدول بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Edit Query In DataSet لعرض مخطط مجموعة البيانات.
- اضغط الاستعلام FillByPublisher في موصل جدول الكتب بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Configure لعرض نافذة تحرير الاستعلام.
- اضغط Next لعرض نافذة وسائل الاستعلام، وضع علامة الاختيار أمام Return a DataTable، وغير اسم الوسيلة إلى GetDataByPublisher واضغط Finish.

أيضا، سيضاف رف أدوات ToolStrip إلى النموذج، عليه لافتة تحمل الاسم Publisher (وهو اسم المعامل المراد إدخاله لتنفيذ الاستعلام) ومربع نص ليكتب فيه المستخدم اسم الناشر، وزرا يحمل اسم الوسيلة FillByPublisher، وعند الضغط عليه سيتم ملء مجموعة البيانات بكتب هذا الناشر، فالكود الذي يفعل هذا تم إنتاجه آليا في حدث ضغط الزر.. لكن سيتبقى عليك أن تعرض محتويات مجموعة البيانات للمستخدم، وقد فعلنا هذا بعرضها في جدول عرض كما هو موضح في الصورة:




وتستطيع تغيير عنوان اللافتة والزر، وعرض رف الأدوات من اليمين إلى اليسار

والمشروع TableAdapter يريك أمثلة على استخدام موصلات الجداول، مع استخدام مدير الموصلات في حدث ضغط الزر "حفظ التغييرات" لإرسال التغييرات التي أجراها المستخدم على السجلات إلى قاعدة البيانات.

# -13-

# الجداول والعلاقات والقيود

سنتعرف في هذا الفصل على الكائنات الداخلة في تكوين مجموعة البيانات DataSet، وهي:

- كائن الجدول DataTable والكائنات الداخلية المكونة له مثل كائن الصف DataRow وكائن العمود DataCoulmn.
- كائن العلاقة DataRelation.
- كائنات القيود Constraints المختلفة.

إضافة إلى المجموعات التي تستخدمها هذه الفئات.

## فئة أساس مجموعة البيانات الداخلية
## InternalDataCollectionBase Class

هذه الفئة تمثل واجهة المجموعة ICollection، ولا تزيد على خصائصها ووسائلها بشيء جديد.
وهذه الفئة هي الفئة الأم لكل من المجموعات التالية:

1. DataTableCollection Class
2. DataColumnCollection Class
3. DataRowCollection Class
4. DataRelationCollection Class
5. ConstraintCollection Class

## فئة مجموعة الجداول DataTableCollection Class

هذه المجموعة ترث الفئة InternalDataCollectionBase، وهي تحتوي على عناصر من النوع DataTable.. ويمكن الحصول على هذه المجموعة باستخدام الخاصيّة DataSet.Tables.

والكود التالي يعرض كلّ أسماء الجداول الموجودة في مجموعة البيانات Ds:

```
For Each Tbl As DataTable In Ds.Tables
    MsgBox(tbl.TableName)
Next
```

ولا تضيف هذه المجموعة جديدا إلى خصائص ووسائل واجهة المجموعة ICollection، ولكنها تضيف بعض الصيغ إلى بعض هذه العناصر، مثل:

### العنصر Item:

هذه هي الخاصية الافتراضية، وهي تعيد كائن البيانات DataTable الموجود في موضع معين في القائمة.. ولهذه الخاصية ثلاث صيغ:

1- الصيغة الأولى تستقبل رقم الجدول في المجموعة.

2- والصيغة الثانية تستقبل نصا يمثل اسم الجدل.

3- والصيغة الثالثة تزيد على الصيغة السابقة بمعامل ثان، يستقبل نصا يمثل اسم النطاق Name Space الذي يوجد تحته الجدول في مجموعة البيانات.

وتعيد هذه الوسيلة Nothing إذا لم تجد الجدول المطلوب في المجموعة.

### إضافة Add:

تضيف جدولا إلى مجموعة البيانات، ولها أربع صيغ:

1- الصيغة الأولى بدون معاملات، وهي تنشئ جدولا باسم افتراضيّ (Table1 أو Table2 وهكذا...) وتضيفه إلى مجموعة البيانات.

2- الصيغة الثانية تستقبل معاملا نصيّا، هو اسم الجدول الذي سيتمّ إنشاؤه وإضافته إلى المجموعة.. ولو أرسلت إلى هذا المعامل نصا فارغا ""، فسيسمى الجدول بالاسم الافتراضي Table1 أو Table2 وهكذا... لاحظ أن إضافة جدول بنفس اسم جدول موجود سابقا سيؤدي إلى حدوث خطأ.

3- الصيغة الثالثة تزيد على الصيغة السابقة بمعامل ثان، يوضح نطاق الاسم الذي سيضاف إليه الجدول داخل مجموعة البيانات.. هذا يتيح لك إضافة أكثر من جدول بنفس الاسم إلى مجموعة الجداول، لكن كلا منها ينتمي إلى نطاق مختلف.. هذا مفيد عندما تملأ مجموعة البيانات بجداول متشابهة الأسماء من أكثر من قاعدة بيانات.. مثلا:

```
Ds.Tables.Add("MyTable", "Db1")
Ds.Tables.Add("MyTable", "Db2")
```

4- والصيغة الرابعة تستقبل كائن جدول DataTable، لتتم إضافته إلى المجموعة.

لاحظ أن الصيغ الثلاث الأولى تعيد كائنا من النوع DataTable يمثل الجدول الذي تمّ إنشاؤه، بينما الصيغة الرابعة هي إجراء لا يعيد أية قيمة، وذلك لأنك أرسلت إليها كائن الجدول بالفعل، ولا تحتاج إلى مرجع آخر له.

**يمكن حذفه CanRemove:**

تعيد True إذا كان من الممكن حذف كائن الجدول DataTable المرسل كمعامل من مجموعة الجداول.. وتعيد هذه الوسيلة False إذا لم يكن الجدول موجودا في المجموعة، أو كان داخلا في علاقة.. والكود التالي سيعيد False لأن حذف جدول المؤلفين سيدمر تكامل العلاقة مع جدول الكتب:

```
MsgBox(Ds.Tables.CanRemove(Ds.Tables("Authors")))
```

**حذف Remove:**

تحذف الجدول المرسل إليها كمعامل، ولها نفس صيغ الوسيلة Add ما عدا الصيغة الأولى التي بدون معاملات.. لاحظ أنك لا تستطيع حذف جدول من مجموعة البيانات إذا كان داخلا في علاقة، فهذا سيؤدي إلى حدوث خطأ في البرنامج.. والمقصود بالعلاقة هنا، العلاقة المعرفة في مجموعة البيانات، فلو وضعت جدولي المؤلفين والكتب في مجموعة البيانات بدون إنشاء علاقة بينهما، فسيمكنك حذف أيهما بدون مشاكل رغم أن هناك علاقة بينهما فعلا في قاعدة البيانات.. أما لو أنشأت العلاقة بينهما في مجموعة البيانات، فعليك حذفها أولا قبل محاولة حذف أي من الجدولين.. والأفضل استخدام الوسيلة CanRemove أولا قبل محاولة حذف الجدول.. مثال:

```
Dim T = Ds.Tables("Authors")
If Ds.Tables.CanRemove(T) Then
  Ds.Tables.Remove(T)
End If
```

**تحتوي على Contains:**

تعيد True إذا كان الجدول المرسل كمعامل موجودا في مجموعة البيانات.. ولهذه الوسيلة صيغتان، تماثلان الصيغتين الثانية والثالثة للوسيلة Add.

**رقم العنصر IndexOf:**

تعيد رقم الجدول المرسل إليها كمعامل إذا كان موجودا في المجموعة، وتعيد 1- إن لم يكن موجودا، ولها نفس صيغ الوسيلة Add ما عدا الصيغة الأولى التي بدون معاملات.

كما تمتلك مجموعة الجداول حدثين جديدين، هما:

**المجموعة تتغير CollectionChanging:**

ينطلق عندما توشك جداول المجموعة على التغير، نتيجة إضافة أو حذف جدول.. والمعامل الثاني e لهذا الحدث من النوع CollectionChangeEventArgs، وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| | Action | تعيد إحدى قيم المرقم CollectionChangeAction التي توضح نوع الفعل الذي سبب تغير المجموعة.. وهذه القيم هي:<br>– Add: إضافة عنصر إلى المجموعة.<br>– Remove: حذف عنصر من المجموعة.<br>– Refresh: تغير عناصر المجموعة كلها، بسبب بعض الوسائل مثل Clear التي تمحو كل العناصر. |
| | Element | تعيد كائنا Object يحتوي على العنصر الذي تعرض للتغيير.. لاحظ أن قيمة هذا العنصر ستكون Nothing إذا كانت للخاصية Action القيمة Refresh. |

**المجموعة تغيرت CollectionChanged:**

ينطلق بعد حدوث التغيير فعليا في عناصر المجموعة.. والمعامل الثاني e لهذا الحدث من النوع CollectionChangeEventArgs الذي تعرفنا عليه في الحدث السابق.

## فئة جدول البيانات DataTable Class

تعمل هذه الفئة كوعاء لأحد الجداول بما فيه من أعمدة وصفوف، وهي تمثل الواجهة IListSource، كما أنها ترث الفئة MarshalByValueComponent، مما يتيح لك إضافتها إلى صينية مكونات النموذج، وإن كان عليك أن تضيفها أولا إلى صندوق الأدوات.. لكن لا داعي لهذا، فأنت تستطيع التعامل مع الجدول بطريقة مرئية في وقت التصميم، بعرض خصائص مجموعة البيانات غير محددة النوع Un-typed DataSet في نافذة الخصائص، واستخدام الخاصية Tables لإضافة الجداول وتغيير خصائصها بطريقة مرئية.. أما إذا كنت تتعامل مع مجموعة بيانات محددة النوع Typed DataSet، فيمكنك إضافة الجداول مباشرة إلى مخطط XML بالطرق التي تعرفنا عليها في الفصل السابق.

ولحدث إنشاء هذه الفئة أربع صيغ:

1- الصيغة الأولى بدون معاملات.

2- والصيغة الثانية تستقبل معاملا نصيّا، هو اسم الجدول.

3- والصيغة الثالثة تزيد على الصيغة السابقة بمعامل ثان، يوضح نطاق الاسم الذي سيضاف إليه الجدول داخل مجموعة البيانات.

4- والصيغة الرابعة تستقبل معاملين من النوعين SerializationInfo و StreamingContext لاستخدامها في سلسلة الجدول Serialization.. هذا الموضوع خارج نطاق هذا الكتاب.

والمثال التالي يعرف كائن جدول ويضع فيه أول جدول في مجموعة جداول مجموعة البيانات:

```
Dim T As DataTable = Ds.Tables(0)
```

والمثال التالي يعرف كائن جدول جديد ويضيفه إلى مجموعة البيانات:

```
Dim T As New DataTable("MyTable")
Ds.Tables.Add(T)
```

ويمتلك كائن الجدول الخصائص التالية:

### اسم الجدول TableName:

تقرأ أو تغير اسم الجدول في مجموعة البيانات.

### نطاق الاسم Namespace:
تحدد اسم النطاق الذي سيندرج تحته الجدول.. هذا يسمح بوجود أكثر من جدول بنفس الاسم في مجموعة البيانات، إذا كان كل منها في نطاق مختلف.

### البادئة Prefix:
تحدد البادئة التي ستميز الجدول كاختصار لاسم نطاقه.

### مجموعة البيانات DataSet:
تعيد كائن مجموعة البيانات DataSet التي ينتمي إليها هذا الجدول.. وتعيد Nothing إذا لم يكن الجدول مضافا إلى مجموعة بيانات حاليا.

### تنسيق التعامل عن بعد RemotingFormat:
تحدد التنسيق الذي سيتم به إرسال بيانات الجدول من جهاز إلى آخر، عندما التعامل مع برنامج يستخدم التحكم عن بعد Remoting، وهي تأخذ إحدى قيمتي المرقم SerializationFormat اللتين تعرفنا عليهما سابقا.

### حساس لحالة الأحرف CaseSensitive:
لو جعلت هذه الخاصيّة True، فستتمّ مراعاة حالة الحروف (صغيرة Small أو كبيرة Capital) عند مقارنة النصوص في هذا الجدول.

### المحل Locale:
تقرأ أو تغير كائن معلومات الثقافة CultureInfo، الذي يمثل اللغة التي تريد استخدامها لمقارنة وترتيب النصوص.

### أقل سعة MinimumCapacity:
تحدد أقل عدد من السجلات يمكن أن يحتويه الجدول.. والقيمة الافتراضية لهذه الخاصية هي 50 سجلا.. وتفيدك هذه الخاصية في حجز الذاكرة اللازمة عند التعامل مع جداول ضخمة الحجم، لتحسين كفاءة البرنامج، فأنت تحدد سعة مبدئية كبيرة لحجز الذاكرة المناسبة، بدلا من أن يتم حجز ذاكرة صغيرة، ثم يضطر البرنامج إلى زيادتها أكثر من مرة أثناء تشغيله.

### الأعمدة Columns:

تعيد مجموعة الأعمدة DataColumnCollection الموجـودة فـي هـذا الجدول.. وسنتعرف على هذه المجموعة بالتفصيل لاحقا.

### الصفوف Rows:

تعيد مجموعة الـصفوف DataRowCollection الموجـودة فـي هـذا الجدول.. وسنتعرف على هذه المجموعة بالتفصيل لاحقا.

### العلاقات الرئيسية ParentRelations:

تعيد نسخة من مجموعة العلاقات DataRelationCollection تحتوي على العلاقات الخارجة من هذا الجدول (العلاقات التي يدخل فيها كجدول رئيسيّ Master Table).. وسنتعرف علـى الفئـة DataRelationCollection بالتفصيل لاحقا.

### العلاقات الفرعية ChildRelations:

تعيد نسخة من مجموعة العلاقات DataRelationCollection، تحتـوي على العلاقات القادمة إلى هذا الجدول (العلاقات التي يدخل فيهـا كجـدول فرعي أو جدول التفاصيل Details Table).

### المفتاح الأساسي PrimaryKey:

تستقبل هذه الخاصية مصفوفة أعمدة DataColumn Array، تحتوي على الأعمدة التي تريد استخدامها كمفتاح أساسي للجدول.. ويمكنـك اسـتخدام مصفوفة بها خانة واحدة إذا كان المفتاح الأساسي يتكون من عمود واحد، أو استخدام مصفوفة بها أكثر من خانة إذا كنت تستخدم عمودين أو أكثر معـا كمفتاح رئيسيّ للجدول.. مثلا: لو كان لديك جدول به عمـود للاسـم الأول للشخص، وعمود آخر لاسمه الأوسط، وعمود ثالث لاسمه الأخيـر، فكـل عمود من هذه الأعمدة لا يصلح بمفرده كمفتاح أساسي بسبب تكرر الأسماء به، بينما قد تصلح الأعمدة الثلاثة معا كمفتاح أساسي، لأن الاسـم الثلاثـي نادرا ما يتكرر.. كل ما عليك في هذه الحالة هو وضع كائنات هذه الأعمدة في مصفوفة ووضعها في هذه الخاصية، لتـصير هـذه الأعمـدة المفتـاح الأساسي.

### القيود Constraints:

تعيد مجموعة القيود ConstraintCollection الموجودة في هذا الجدول.. وسنتعرف على هذه المجموعة بالتفصيل لاحقا.

### العرض الافتراضي DefaultView:

تعيد كائن العرض DataView Object الذي يحمل مبدئيا كل بيانات الجدول الحالي، لكنك تستطيع ضبطه لعرض جزء فقط من سجلات الجدول تبعا لشرط معين.. وسنتعرف على فئة عرض البيانات DataView Class بالتفصيل في الفصل التالي.

### تعبير العرض DisplayExpression:

تقرأ أو تغير النصّ الذي سيتمّ عرضه للمستخدم كعنوان للجدول في أدوات عرض البيانات كالأداة DataGridView.

### به أخطاء HasErrors:

تعيد True إذا كانت هناك أيّة أخطاء في أي صفّ في هذا الجدول.

### الخصائص الإضافية ExtendedProperties:

تعيد مجموعة الخصائص PropertyCollection التي تحتوي على الخصائص الإضافيّة للجدول.. والمثال التالي يضيف خاصيّة اسمها Password إلى جدول الكتب، ويضع فيها القيمة "كلمة المرور"، ثم يغيرها إلى "أحمد123":

```
Dim T As DataTable = Ds.Tables("Books")
Dim EP As PropertyCollection = T. ExtendedProperties
EP.Add ("Password", "كلمة المرور")
EP.Item("Password") = "أحمد123"
MsgBox(EP("Password"))
```

كما يمتلك كائن الجدول الوسائل التالية:

### نسخ Clone:
تعيد كائن جدول DataTable جديدا، وتنسخ إليه مخطط الجدول الحالي Schema بكلّ ما فيه من أعمدة وقيود.. ولكنّ الجدول الناتج يكون فارغا من السجلات.

### نسخ Copy:
تعيد كائن جدول DataTable جديدا، وتنسخ إليه الجدول الحالي بمخططه وسجلاته، ليكون مماثلا للجدول الأصلي تماما.

### محو Clear:
تمحو كلّ السجلات الموجودة في الجدول.

### تصفير Reset:
تفرغ الجدول تماما من كل أعمدته وسجلاته وقيوده.

### صف جديد NewRow:
تنشئ سجلا جديدا له نفس مخطط الجدول (نفس الأعمدة بنفس أنواع بياناتها بنفس ترتيبها)، وتعيد إليك كائن الصف DataRow الذي يشير إلى هذا السجل، لكن دون إضافته إلى مجموعة صفوف الجدول Rows، لهذا عليك أن تضيفه إليها بنفسك.. مثال:

```
Dim T As DataTable = Ds.Tables("Authors")
Dim R = T.NewRow
R("Author") = "أحمد شوقي"
R("About") = "أمير الشعراء"
R("CountryID") = 21
T.Rows.Add(R)
```

### استعارة صف ImportRow:
أرسل إلى هذه الوسيلة كائن صف DaraRow، لتنسخه وتضيفه إلى الجدول بكلّ بياناته وخصائصه، بما في ذلك النسخة الأصلية Original Version والنسخة الحالية Current Version لقيم خاناته.

### تحميل صف LoadDataRow:

استخدم هذه الوسيلة لتحديث أحد سجلات الجدول، أو إضافة سجل جديد إليه.. وتستقبل هذه الوسيلة معاملين:

- مصفوفة كائنات Object Array، تحتوي على قيم خانات السجل بنفس ترتيبها في الجدول.. لاحظ أن هذه الوسيلة ستبحث في الجدول، لترى إن كان المفتاح الأساسي لأي حقل له نفس القيمة الموجودة في الخانة المناظرة في المصفوفة.. فإذا كان المفتاح الأساسي موجودا في الجدول، يتم نسخ باقي القيم من المصفوفة إلى باقي حقول السجلّ لتحديثها.. وإذا لم يكن المفتاح الأساسي موجودا، يتم إنشاء سجلّ جديد وتوضع بحقوله قيم المصفوفة.
لاحظ أن ترك إحدى خانات المصفوفة فارغة، سيؤدي إلى وضع القيمة الافتراضية في العمود المناظر لها إن كانت له قيمة افتراضية، أو سيتم توليد الترقيم التلقائي إذا كانت للخاصية AutoIncrement لهذا العمود القيمة True.. فإذا لم يكن هذا أو ذاك، وكانت الخانة لا تقبل أن تظل فارغة، فسيحدث خطأ في البرنامج.. ويحدث خطأ أيضا إذا كان عدد خانات المصفوفة أكبر من عدد أعمدة الجدول.
- معامل منطقي Boolean، لو جعلت قيمته True فسيتمّ استدعاء الوسيلة AcceptChanges بعد إضافة السجل إلى الجدول وبهذا يعتبر هذا السجل سجلا أصليا لم يحدث له أي تغيير.. أما إذا جعلت قيمة هذا المعامل False، فسيعتبر السجل الجديد سجلا مضافا Added ويعتبر السجل الذي تم تحديثه سجلا معدلا Modified.

وتعيد هذه الوسيلة كائن صفّ DataRow يحمل مرجعا إلى الصفّ الذي تمّ تحديثه أو إضافته.

| ملحوظة: |
| --- |
| تقوم فئة المجموعة المحددة النوع Typed DataSet Class، بتعريف عدة وسائل محددة النوع في كل جدول للتعامل مع صفوفه.. على سبيل المثال، لو كان في المجموعة محددة النوع فئة لجدول المؤلفين اسمها AuthorsDataTable، وتم تعريف فئة اسمها AuthorsRow تمثل نوع صفوف هذا الجدول، فإن هذا الجدول سيحتوي على الوسائل التالية: |

**صف مؤلفين جديد NewAuthorsRow:**

تعيد كائنا من النوع AuthorsRow يمثل صفا جديدا من صفوف جدول المؤلفين، بحيث يمكنك إضافته إلى جدول المؤلفين.. مثال:

```
Dim R = Ds.Authors.NewAuthorsRow
R.Author = "أحمد شوقي"
R.About = "أمير الشعراء"
R.CountryID = 21
Ds.Authors.AddAuthorsRow(R)
```

**إضافة صف المؤلفين AddAuthorsRow:**

تستقبل هذه الوسيلة معاملا من النوع AuthorsRow يمثل صفا من صفوف جدول المؤلفين، لإضافته إلى جدول المؤلفين، كما رأينا في المثال السابق.

وتوجد صيغة أخرى لهذه الوسيلة تستقبل قيم صف المؤلفين لإضافته إلى الجدول في خطوة واحدة.. هكذا مثلا يمكن اختزال المثال السابق:

```
Ds.Authors.AddAuthorsRow("أحمد شوقي", 21, "",
            "أمير الشعراء", Nothing)
```

لاحظ أننا أرسلنا نصا فارغا إلى خانة رقم الهاتف، كما وضعنا القيمة Nothing في خانة طابع الوقت لأن قاعدة البيانات تولدها تلقائيا.. أما المفتاح الرئيسي لهذا الصف (وهو الحقل ID) فلم تطالبنا به هذه الوسيلة أصلا لأنها تعرف أنه يولد تلقائيا.

**حذف صف المؤلفين RemoveAuthorsRow:**

تستقبل هذه الوسيلة معاملا من النوع AuthorsRow يمثل صفا من صفوف جدول المؤلفين، لحذفه من جدول المؤلفين.

**تحديد Select:**

تعيد مصفوفة صفوف DataRow Array، تحتوي على بعض أو كلّ صفوف الجدول.. ولهذه الوسيلة الصيغ التالية:

1- الصيغة الأولى بدون معاملات، وهي تعيد مصفوفة تحتوي على كل سجلات الجدول.

2- الصيغة الثانية تستقبل معاملا نصّيا، يمثل الشرط الذي على أساسه سيتم اختيار السجلات من الجدول، ويمكنك صياغة هذا الشرط بنفس

قواعد صياغة الفقرة WHERE في استعلامات SQL.. والجملة التالية تعيد كلّ الكتب التي تبدأ بحروف تسبق حرف الثاء في الترتيب الأبجديّ:

**Dim R( ) As DataRow = T.Select ("Book < 'ث' ")**

ويمكنك أن تستخدم في تكوين الشرط، الدوال والكلمات المستخدمة في تكوين شرط الخاصية DataRow.Expression التي سنتعرف عليها لاحقا.

3- الصيغة الثالثة تزيد على الصيغة السابقة بمعامل نصي يحدد ترتيب الصفوف.. ويتكون هذا المعامل من شقين:

أ- اسم العمود الذي يتم الترتيب على أساسه (مثل Book)، أو أي تعبير يجمع بين أكثر من عمودين كناتج ضربهما (مثل Copies_No * Price).

ب-نوع الترتيب، وهو إحدى الكلمتين التاليتين:

- ASC: للترتيب التصاعديّ وهو الترتيب الافتراضيّ لهذا يمكن ألا تكتب هذه الكلمة.
- DESC: للترتيب التنازلي.

والمثال التالي يعرض أسماء الكتب التي تبدأ بحروف تسبق حرف الثاء في الترتيب الأبجديّ، مرتبة تنازليّا على حسب اسم الكتاب:

**Dim R = T.Select ("Book < 'ث' ", "Book DESC")**

4- الصيغة الرابعة تزيد على الصيغة السابقة بمعامل ثالث من نوع المرقم DataViewRowState، مما يتيح لك تحديد حالة السجلات التي تريد تطبيق الشرط عليها.. هذا يمكنك من البحث في السجلات المضافة أو المعدّلة أو المحذوفة... إلخ.. وسنتعرّف على المرقم DataViewRowState بالتفصيل لاحقا.

**حساب Compute:**

تبحث في الجدول عن السجلات التي تحقق الشرط المرسل إلى المعامل الثاني، وتجري على هذه السجلات دالة التجميع Aggregate Function المرسلة إلى المعامل الأول.. وتعيد هذه الوسيلة كائنا Object يحمل ناتج عملية التجميع.. دعنا نأخذ مثالا: افترض أنك تريد حساب عدد الكتب التي تبدأ بحروف تسبق حرف النون في جدول الكتب.. يمكنك فعل هذا باستخدام الوسيلة Compute كالتالي:

**C = TblBooks.Compute("Count(Book)", "Book < 'ن' ")**

لاحظ أنك تستطيع إرسال نص فارغ إلى المعامل الثاني، وفي هـذه الحالـة سيتم تطبيق دالة الجميع على جميع سجلات الجدول.. والجملة التالية تعيـد إليك مجموع نسخ الكتب في الجدول:

```
C = TblBooks.Compute("Sum(Copies_No)", "")
```

ويوجد عيب خطير في هذه الوسيلة، فهي لا تستطيع حساب دالـة التجميـع على أكثر من عمود مباشرة.. فإذا أردت مثلا أن تحسب مجموع أثمان كـل نسخ الكتب الموجودة في الجدول، فإن الجملة التالية غير مقبولة:

```
C = TblBooks.Compute("Sum(Copies_No * Price)", "")
```

ولحلّ هذه المشكلة، عليك إنشاء عمود جديد في الجدول، واستخدام خاصـيّة "الصيغة" Expression الخاصّة به لتكون قيم خاناته هي حاصـل ضـرب العمودين المطلوبين، ثم تجري على هذا العمود الحسابات التـي تريـدها.. وسنتعرف الأعمدة المحسوبة بالتفصيل عند التعرف على خـصائص كـائن العمود DataCoulmn.

وقد استخدمنا هذه الوسيلة في حدث ضغط زر تحميل البيانات في المشروع CustomDataSet لحساب مجموع درجات كل طالب وعرضه في عمود "المجموع" في جدول العرض.. السبب في هذا أن مجموع درجات كل طالب لا يتم حفظه في الملف، لان عمود "المجموع" مضاف إلى جدول العـرض فقط وليس موجودا في مجموعة البيانات، لهـذا علينـا أن نحـسب قيمتـه بأنفسنا.. هذا هو الكود الذي يفعل هذا:

```
SumCell.Value = DsStudents.Grades.Compute(
    "SUM(Grade)", "StudentID = " & Trim(StdId))
```

حيث StdId هو متغير يحمل رقم الطالب المراد حساب مجموع درجاته.

### معرفة التغييرات GetChanges:

تعيد كائن جدول DataTable جديدا، يحتوي فقط على الصفوف التي تـمّ تعديلها أو إضافتها أو حذفها من الجدول الحاليّ، منذ أن تمّ تحميله أو منـذ آخر استدعاء للوسيلة AcceptChanges.

وهناك صيغة ثانية لهذه الوسـيلة، تـستقبل معـاملا مـن نـوع المـرقم DataRowState الذي تعرفنا عليه من قبل، لتتمكن من اختيار الـسجلات التي وقع عليها نوع معين من التغيير دون غيره.

- **معرفة الأخطاء GetErrors:**

تعيد مصفوفة صفوف DataRow Array، تحتوي على الصفوف التي حدثت بها أخطاء عند محاولة حفظ الجدول في قاعدة البيانات، ليمكنك تصحيح أخطائها.

- **قبول التغييرات AcceptChanges:**

تقوم باستدعاء الوسيلة AcceptChanges الخاصة بكل صف في الجدول.

- **رفض التغييرات RejectChanges:**

تقوم باستدعاء الوسيلة RejectChanges الخاصة بكل صف في الجدول.

- **بدء تحميل البيانات BeginLoadData:**

يؤدي استدعاء هذه الوسيلة إلى:

- إيقاف إرسال التنبيهات Notifications إلى فئات ADO.NET التي تتعامل مع الجدول.. هذا معناه إيقاف انطلاق الأحداث Events الخاصة بهذه الفئات.
- إيقاف تحديث الفهارس Indexes.
- إيقاف التحقق من قواعد الصحّة Constraints.

وعليك استدعاء هذه الوسيلة قبل البدء في إضافة عدد كبير من السجلات إلى الجدول، لأن تكرار تنفيذ العمليات المذكورة سابقا بعد إضافة كل سجل إلى الجدول يستهلك وقتا ملموسا ويجعل البرنامج بطيئا، لهذا من الأذكى إيقافها مؤقتا، ثم إعادة تشغيلها بعد الانتهاء من ملء الجدول بالسجلات.

- **انتهاء تحميل البيانات EndLoadData:**

هذه الوسيلة رديفة للوسيلة BeginLoadData، وعليك استدعاؤها بعد انتهاء تحميل السجلات في الجدول، لإعادة تشغيل العمليات الخاصة بالتنبيهات والأحداث والفهارس والقيود، وبهذا تضمن تنفيذها مرة واحدة فقط بعد عملية التحميل.

- **دمج Merge:**

تضيف سجلات الجدول المرسل إليها كمعامل، إلى الجدول الحالي، وإن كانت السجلات المضافة موجودة سابقا، يتم تحديث السجلات الموجودة بقيم السجلات القادمة.

ولهذه الوسيلة نفس صيغ الوسيلة DataSet.Merge، مع اختلاف واحد، هو أن المعامل الأول في هذه الصيغ هو كائن جدول بيانات DataTable وليس DataSet.

## إنشاء قارئ بيانات CreateDataReader:

تعيد قارئ بيانات الجدول DataTableReader، الذي يمكنك من خلاله المرور عبر كل سجلات الجدول الحالي.. وسنتعرف على الفئة DataTableReader لاحقا في هذا الفصل.

## كتابة كود المخطط WriteXmlSchema:

تكتب كود XML الذي يعبر عن مخطط الجدول الحالي، في الملف المرسل كمعامل، سواء كان هذا المعامل في صورة نص يمثل اسم الملف، أو كائن يتعامل مع الملف من الأنواع Stream أو TextWriter أو XmlWriter.
وتوجد صيغة ثانية لهذه الوسيلة، تزيد عليها بمعامل منطقي، إذا جعلت قيمته True فسيتم حفظ مخطط الجداول الفرعية التابعة لهذا الجدول أيضا مع مخطط الجدول الحالي.

## كتابة الكود WriteXml:

تكتب كود XML الذي يمثل سجلات الجدول، في الملف المرسل كمعامل، سواء كان هذا المعامل في صورة نص يمثل اسم الملف، أو كائن يتعامل مع الملف من الأنواع Stream أو TextWriter أو XmlWriter.
ولبعض صيغ هذه الوسيلة معامل منطقي، إذا جعلت قيمته True فسيتم حفظ سجلات الجداول الفرعية التابعة لهذا الجدول أيضا.
ولبعض صيغ هذه الوسيلة معامل من نوع المرقم XmlWriteMode الذي تعرفنا عليه سابقا، لتستطيع من خلاله اختيار طريقة حفظ البيانات.

## قراءة كود المخطط ReadXmlSchema:

تقرأ مخطط جدول أو أكثر من الملف المرسل كمعامل، سواء كان هذا المعامل في صورة نص يمثل اسم الملف، أو كائن يتعامل مع الملف من الأنواع Stream أو TextWriter أو XmlWriter.. وتضيف هذه الوسيلة هذه المخططات إلى الجدول الحالي والجداول الفرعية التابعة له.

- **قراءة الكود ReadXml:**
تقرأ سجلات جدول أو أكثر، من الملف المرسل كمعامل، سواء كان هذا المعامل في صورة نص يمثل اسم الملف، أو كائن يتعامل مع الملف من الأنواع Stream أو TextWriter أو XmlWriter.. وتضيف هذه الوسيلة هذه السجلات إلى الجدول الحالي والجداول التابعة له.

كما يمتلك كائن الجدول الأحداث التالية:

- **العمود تغير ColumnChanged:**
ينطلق بعد أن تتغيّر إحدى القيم في أحد أعمدة الجدول.. والمعامل الثاني e لهذا الحدث من النوع DataColumnChangeEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | Column | تعيد كائن العمود DataColumn الذي حدث به التغيير. |
| | Row | تعيد كائن الصفّ DataRow، الذي توجد به الخانة التي تغيرت. |
| | ProposedValue | تقرأ أو تغير القيمة المقترحة (التي تغيرت).. هذا يتيح لك ـ لو أردت ـ تعديل القيمة التي تغيرت. |

- **العمود يتغير ColumnChanging:**
مماثل للحدث السابق، ولكنه ينطلق عند محاولة إجراء التغيير في أحد أعمدة الجدول (أي قبل حدوث التغيير بالفعل).. ويمكنك استخدام الكود التالي لإلغاء تغيير قيمة الخانة:

**e.ProposedValue = e.Row(e.Column)**

لكن عليك استخدام هذه الجملة داخل شرط، فلو استخدمتها هكذا بمفردها فستمنع المستخدم من تغيير أي خانة في أي عمود في الجدول.. لهذا فالعملي أن تستخدمها لمنع بعض القيم الخاطئة، مثل ترك عمود اسم المؤلف فارغا لأن هذا غير مقبول في قاعدة البيانات.. وستجد هذا الكود في الفئة الجزئية لجدول المؤلفين في مجموعة البيانات محددة النوع في المشروع TableAdapter:

```
If e.Column Is AuthorColumn AndAlso
   e.ProposedValue = "" Then
      e.ProposedValue = e.Row(e.Column)
End If
```

ويمكنك بنفس الطريقة، إضافة أي شروط أخرى للتأكد من صلاحية القيم التي يدخلها المستخدم في باقي الأعمدة.

### الصف تغير RowChanged:

ينطلق بعد أن تتغيّر إحدى القيم في أحد سجلات الجدول.. والمعامل الثاني e لهذا الحدث من النوع DataRowChangeEventArgs، وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| | Row | تعيد كائن الصفّ DataRow، الذي توجد به الخانة التابعة للعمود الذي حدث به التغيير. |
| | Action | تخبرك بنوع التغيير الذي حدث للصفّ، وهي تعيد إحدى قيم المرقم DataRowAction التالية:<br>- Add: تمت إضافة الصف.<br>- Delete: تم حذف الصف.<br>- Change: يتم تغيير إحدى قيم الصف.<br>- ChangeOriginal: تم تغيير النسخة الأصلية Original Version من السجل.<br>- ChangeCurrentAndOriginal: تم تغيير النسخة الأصلية Original Version والنسخة الحالية Current Version من السجل.<br>- Commit: تم نقل التغييرات التي حدثت على تعاملات الصف Transactions إلى قاعدة البيانات نهائيا.<br>- Rollback: تم التراجع عن التغييرات التي حدثت على تعاملات الصف.<br>- Nothing: لم يحدث أي تغيير على الصف. |

### الصف يتغير RowChanging:

مماثل للحدث السابق، ولكنّه ينطلق عند محاولة إجراء التغيير في أحد سجلات الجدول (قبل حدوث التغيير).

### صف جديد للجدول TableNewRow:

ينطلق بعد استدعاء الوسيلة NewRow التي تعيد سجلا جديدا من سجلات الجدول.. والمعامل الثاني e لهذا الحدث من النوع DataTableNewRowEventArgs، وهو يمتلك الخاصية Row التي تعيد كائن الصف DataRow الذي يمثل الصف الجديد الذي تم إنشاؤه.. هذا يتيح لك وضع أية قيم افتراضية تريها في خانات الصف الجديد.

### تم حذف الصف RowDeleted:

ينطلق بعد حذف أحد سجلات الجدول.. والمعامل الثاني e لهذا الحدث من النوع DataRowChangeEventArgs وقد تعرفنا عليه سابقا.

لاحظ أن هذا الحدث ينطلق والصف ما زال موجودا فعلا في مجموعة صفوف الجدول لكن حالته تكون DELETED، وهو ما سيسبب خطأ في البرنامج لو حاولت إضافة الصف مرة أخرى إلى الجدول!!

### يتم حذف الصف RowDeleting:

مماثل للحدث السابق، ولكنه ينطلق عند محاولة حذف أحد سجلات الجدول (قبل إتمام الحذف).. والحقيقة أن هذا الحدث قليل الفائدة، لأنه لا يمتلك خاصية لإلغاء عملية الحذف قبل وقوعها، وكان المنتظر أن يمتلك هذا الحدث الخاصية e.Cancel لتجعله مفيدا!

وكل ما تستطيع فعله فيه، هو حفظ الصف الذي سيتم حذفه في متغير احتياطي، ليمكنك التراجع عن عملية الحذف بعد هذا لو أردت.

لاحظ أنك لا تستطيع حذف الصف بنفسك من داخل هذا الحدث، فلو حاولت استخدام الوسيلة DataTable.Rows.Remove(e.Row) لحذف الصف، فسيحدث خطأ في البرنامج!

ولو أردت منع عملية الحذف بأبسط طريقة، فعليك فعل هذا من أداة العرض.. مثلا: لو وضعت زرا على النموذج للقيام بعملية الحذف، فيمكنك أن تضيف إليه الكود الذي يسأل المستخدم إن كان يريد إتمام الحذف فعلا أم لا.. أما إن كنت تستخدم جدول عرض، فعليك استخدام الأحداث الخاصة به لفعل هذا، وإلغاء عملية الحذف إن قرر المستخدم هذا.

أما لو كنت مصرا على استخدام هذا الحدث العقيم، فسيتوجب عليك كتابة بعض الكود لفعل هذا.. وقد فعلنا هذا في المشروع التعريف الجزئي لفئة جدول المؤلفين في المشروع TableAdapter، حيث استخدمنا الحدث

AuthorsRowDeleting المشتق من الحدث RowDeleting لسؤال المستخدم إن كان يريد حذف الصف، فإن قرر إلغاء العملية، قمنا بإنشاء نسخة احتياطية من جدول المؤلفين، واستخدام الوسيلة Merge لإضافة نسخة من سجلات الجدول الحالي إلى الجدول الاحتياطي:

```
TempTable = New AuthorsDataTable
TempTable.Merge(Me)
```

ثم استخدام الحدث RowDeleted لدمج سجلات الجدول الاحتياطي بالجدول الأصلي مرة أخرى، وهذا سيعيد حالة السجلات كما كانت، بما في ذلك السجل المحذوف:

```
If TempTable IsNot Nothing AndAlso
        TempTable.Count > 0 Then
   Me.Merge(TempTable)
   TempTable.Clear()
End If
```

لاحظ أن الشرط TempTable.Count > 0 هو المؤشر الذي يشعرنا بأن المستخدم رفض إتمام عملية الحذف.. ولو لم نستخدمه، فسيتم التراجع عن جميع عمليات الحذف بغض النظر عن رأي المستخدم!

والخاصية Count بالمناسبة، هي خاصية معرفة في فئة الجدول محدد النوع، وهي مجرد إختصار للكود التالي:

```
If TempTable.Rows.Count > 0 Then
```

ورغم أنها تعمل بشكل صحيح، يظل بهذه الطريقة عيب خطير، وهو اضطرارنا إلى نسخ كل سجلات الجدول للمحافظة على حالة سجل واحد فقط، وهذه كارثة على الذاكرة وسرعة التنفيذ إذا كان عدد سجلات الجدول ضخما!.. وللأسف، الوسيلة Merge الخاصة بالجدول لا تقبل دمج سجل منفرد، ولا تقبل إلا كائن جدول كمعامل.. ولو أردت حل هذه المشكلة، فعليك استخدام مجموعة بيانات احتياطية، ثم استخدام الوسيلة Merge الخاصة بمجموعة البيانات، لأنها تقبل دمج مصفوفة من السجلات، ومن السهل وضع السجل المراد حذفه في مصفوفة وإرسالها إليها:

```
TempDs = New DsAuthorsBooks
TempDs.Merge({e.Row})
```

| تحذير هام: |
| --- |
| لا تعرف نسخة جديدة من المجموعة TempDs على مستوى فئة جدول المؤلفين.. هذا الكود غير صحيح:<br>**Dim TempDs As New DsAuthorsBooks**<br>السبب في هذا أنه سيؤدي إلى تعريف دائري يعطل البرنامج عن العمل إلى أن يدمر كل مساحة الرصة Stack المتاحة له في الذاكرة.. فعند تعريف نسخة جديدة من مجموعة البيانات DsAuthorsBooks، ستقوم بتعريف نسخة من جدول المؤلفين، التي ستقوم بتعريف نسخة احتياطية من المجموعة TempDs التي ستقوم بتعريف نسخة جديدة من جدول المؤلفين، التي ستقوم بتعريف نسخة احتياطية من المجموعة TempDs وهكذا إلى ما لا نهاية!<br>وهذا نفس ما سيحدث إن استخدمت جدول مؤلفين احتياطيا وعرفت نسخة جديدة منه على مستوى الفئة.<br>ولحل هذه المشكلة، عرف المتغير على مستوى الفئة بدون الكلمة New:<br>**Dim TempDs As DsAuthorsBooks**<br>ثم ضع النسخة الجديدة في هذا المتغير في الحدث RowDeleting:<br>**TempDs = New DsAuthorsBooks** |

لكن.. لماذا لا نستخدم نرفض التغيير الذي حدث للصف المحذوف لنستعيده مباشرة بجملة كالتالية:

```
e.Row.RejectChanges( )
```

فكرة جيدة، لكن هذه الطريقة ستعمل فقط مع المؤلفين القادمين من قاعدة البيانات، أما إذا أضاف المستخدم مؤلفا، ثم قرر حذفه، ثم ضغط Cancel لعدم إتمام عملية الحذف، فستسبب الوسيلة RejectChanges خطأ، لأن السجل المضاف يفقد كل قيمه فعليا عند حذفه!!.. هذا رغم أن هذا الصف ما زال موجودا في الجدول، وتستطيع أن تحصل على رقمه بالكود التالي:

```
MsgBox(Me.Rows.IndexOf(e.Row))
```

لكن حتى لو لم يحدث هذا الخطأ، فسيؤدي إلغاء تعديلات هذا الصف المضاف إلى حذفه من الجدول، وهكذا لن تستعيد الصف في كل الأحوال!
أيضا، لا يمكنك رفض تغييرات الجدول كله:

```
Me.RejectChanges( )
```

لأن هذا سيضيع كل التعديلات التي قام بها المستخدم ولم يحفظها في قاعدة البيانات، كما أنه سيعيد كل السجلات التي حذفها من قبل، وليس فقط آخر سجل محذوف!

لهذا تظل طريقة مجموعة البيانات الاحتياطية أفضل وأدق طريقة يمكن استخدامها.
واضح طبعا أن ميكروسوفت كانت ستحيل حياتنا إلى نعيم لو أضافت الخاصية e.Cancel في هذا الحدث كما هو مألوف!

### تم محو الجدول TableCleared:

ينطلق مباشرة بعد نجاح الوسيلة Clear في محو كل سجلات الجدول، وقبل العودة لتنفيذ باقي الكود الذي استدعى الوسيلة Clear.. لاحظ أن هذا الحدث لن ينطلق إذا حدثت أية أخطاء أثناء حذف سجلات الجدول.
والمعامل الثاني e لهذا الحدث من النوع DataTableClearEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | Table | تعيد كائن الجدول DataTable الذي يتم محو سجلاته. |
| | TableName | تعيد اسم الجدول. |
| | TableNamespace | تعيد نطاق اسم الجدول. |

### يتم محو الجدول TableClearing:

مماثل للحدث السابق، ولكنه ينطلق عند محاولة محو سجلات الجدول (بعد استدعاء الوسيلة Clear لكن قبل تنفيذها).. لاحظ أن هذا الحدث ينطلق دائما، حتى لو كان الجدول فارغا من السجلات فعلا.

## فئة مجموعة الصفوف DataRowCollection Class

هذه الفئة ترث المجموعة InternalDataCollectionBase، وهي تحتوي على عناصر من نوع فئة صف البيانات DataRow Class.
وبخلاف ما ترثه من الفئة الأم، تمتلك هذه المجموعة الوسائل الجديدة التالية:

### إضافة Add:

تضيف صفا إلى مجموعة الصفوف، ولها صيغتان:
1- الصيغة الأولى تستقبل كائن الصف DataRow الذي تريد إضافته.
2- الصيغة الثانية تستقبل مصفوفة كائنات Object Array تحتوي على القيم التي تريد وضعها في خانات السجل.. لاحظ أنك تتعامل مع مجموعة الصفوف من خلال كائن الجدول DataTable، لهذا فإن هذه المجموعة تعرف تركيب السجلات التي ستضيفها إليها، وعليك مراعاة ترتيب الأعمدة وأنواع بياناتها عند وضعها في المصفوفة حتى لا يحدث خطأ، وعليك كذلك ترك خانة المصفوفة المناظرة لخانة الترقيم التلقائي فارغة.
وتقوم هذه الصيغة بإنشاء كائن صف جديد ووضع القيم به وإضافته إلى مجموعة الصفوف، وتعيد إليك كائن صف DataRow يشير إلى الصف الذي تمت إضافته إلى المجموعة.

### تحتوي على Contains:

تعيد True إذا كانت مجموعة الصفوف تحتوي على السجل الذي له المفتاح الأساسي Primary Key المرسل كمعامل.. ولهذا الوسيلة صيغتان:
1- الصيغة الأولى تستقبل كائنا Object يحتوي على قيمة المفتاح الأساسي للسجل الذي تريد البحث عنه.
2- الصيغة الثانية تستقبل مصفوفة كائنات Objects، تحتوي على قيم المفتاح الأساسي، وذلك إذا كان المفتاح الأساسي للجدول يتكون من أكثر من عمود.

### البحث عن Find:

مماثلة للوسيلة السابقة في صيغتيها، إلا أنها تعيد كائن الصف DataRow الذي يملك مفتاحا أساسيا مساويا للقيمة المرسلة كمعامل، وتعيد Nothing إذا لم تعثر على الصف.

وتسبب هذه الوسيلة خطأ في البرنامج إذا لم يكن الجدول الذي تبحث فيه يحتوي على مفتاح أساسي.. يمكن أن يحدث هذا رغم ان الجدول الأصلي في قاعدة البيانات يحتوي على مفتاح أساسي، وذلك إذا استخدمت الوسيلة Fill لملء مجموعة البيانات، دون استخدام الوسيلة FillSchema أولا، فهي التي تنشيء المفتاح الأساسي في جداول مجموعة البيانات.

### الإدراج في موضع InsertAt:

أرسل إلى هذه الوسيلة كائن الصف DataRow والموضع الذي تريد إدراجه فيه في مجموعة الصفوف.

## فئة صفّ البيانات DataRow Class

تتعامل هذه الفئة مع أحد صفوف الجدول، وهي لا تمتلك حدث إنشاء عاما Public Constructor، لهذا لا تستطيع إنشاء نسخة جديدة منها مباشرة، وبدلا من هذا عليك استخدام الوسيلة DataTable.NewRow للحصول على كائن صف جديد.. الحكمة من هذا، هي أن الوسيلة NewRow تستخدم مخطط الجدول لإنشاء صف له نفس الأعمدة بنفس أنواع البيانات ونفس الترتيب.. والكود التالي يعرّف صفا جديدا ويضيفه إلى جدول الكتب:

```
Dim TblBooks = Ds.Tables("Books")
Dim BooksRow As DataRow = TblBooks.NewRow( )
TblBooks.Rows.Add(BooksRow)
```

وتمتلك هذه الفئة الخصائص التالية:

### الجدول Table:

تعيد كائن الجدول DataTable الذي يحتوي على السجلّ الحالي.. تذكر أنك لا تستطيع إنشاء سجل جديد بدون استخدام الوسيلة NewRow من أحد الجداول، لهذا حتى لو لم يكن السجل مضافا إلى مجموعة صفوف الجدول Rows، فإن هذه الخاصية ستظل تشير دائما إلى الجدول الذي تم إنشاء السجل الحالي منه.

### العنصر Item:

هذه هي الخاصية الافتراضية، وهي تقرأ أو تغير القيمة المحفوظة في إحدى خانات السجل الحالي.. لاحظ أن هذه الخاصية من النوع Object، ليمكنك التعامل مع مختلف أنواع الأعمدة.. ولهذه الخاصية الصيغ التالية:

1- بعض الصيغ لها معامل واحد، يستقبل اسم العمود أو رقمه، أو كائن العمود DataColumn الذي يمثله.. والمثال التالي يقرأ اسم المؤلف الموجود في الصفّ الثالث في جدول المؤلفين (لا تنسَ أنّ الصفّ الأول هو الصف رقم صفر):

```
Dim R = Ds.Tables("Authors").Rows.Item(2)
Dim X = R.Item("Author")
```

أو باختصار:

```
Dim R = Ds.Tables("Authors").Rows(2)
Dim X = R("Author")
```

وهو ما يمكنك فعله في سطر واحد كالتالي:

```
Dim X = Ds.Tables("Authors").Rows(2)("Author")
```

حيث يبدو أننا نتعامل مع مجموعة الصفوف Rows كأنها مصفوفة مصفوفات، وذلك لأن الخاصية Rows.Item افتراضية، وكذلك الخاصية Row.Item، مما يمكننا من حذف اسميهما.

أما لو كنت تتعامل مع مجموعة بيانات محددة النوع، فسيختصر الكود السابق إلى:

```
Dim X = Ds.Authors(2)("Author")
```

أو بصورة أفضل:

```
Dim X = Ds.Authors(2).Author
```

2- بعض الصيغ تزيد بمعامل ثان من نوع المرقم DataRowVersion، لتحدد من خلاله نسخة السجل التي تريد التعامل معها.. لكن هذه الصيغ للقراءة فقط، ولا يمكن استخدامها لتغيير قيم السجل.. ويمتلك المرقم DataRowVersion القيم التالية:

| | |
|---|---|
| Original | التعامل مع نسخة السجل الأصلية Original Version. |
| Current | التعامل مع النسخة الحالية للسجل Current Version. |
| Proposed | التعامل مع القيمة المقترحة للسجل.. هذا مفيد إذا كان السجل قيد التحرير ولم يتم إنهاء عملية التحرير بعد، وتريد قراءة القيمة الجديدة قبل قبولها ووضعها في النسخة الحالية Current Version. |
| Default | التعامل مع النسخة الافتراضية للصف، وهي كالتالي:<br>- النسخة الافتراضية للصفوف غير المعدلة هي النسخة الأصلية Original Version.<br>- النسخة الافتراضية للصفوف المعدلة والمضافة والمحذوفة هي النسخة الحالية Current Version.<br>- النسخة الافتراضية للصفوف غير المتصلة بأي جدول Detached، هي النسخة المقترحة Proposed. |

والقيمة المستخدمة مع الصيغ التي لا تمتلك المعامل الثاني هي Default.
انظر المثال التالي:

```
' عرض النسخة الأصلية
MsgBox(Row(0, DataRowVersion.Original))
' عرض النسخة الحالية
MsgBox(Row("Book", DataRowVersion.Current))
' عرض النسخة الافتراضية
MsgBox(Row("Book"))
```

**مصفوفة العنصر ItemArray:**

تعيد مصفوفة كائنات Object Array تحتوي على قيم كلّ خانات السجلّ الحالي بنفس ترتيب الأعمدة.. ويمكنك أن ترسل إليها مصفوفة بها القيم التي تريد وضعها في خانات السجل.. مثال:

```
Dim TblAuthors = Ds.Tables("Authors")
Dim R = TblAuthors.NewRow( )
R.ItemArray = {Nothing, "عنترة بن شداد", 5, "",
                    "شاعر جاهلي", Nothing}
TblAuthors.Rows.Add(R)
```

**به أخطاء HasErrors:**

تعيد True إذا كانت هناك أخطاء متعلقة بالسجل الحالي.. ويمكنك معرفة سبب الخطأ باستخدام الخاصية RowError أو الوسيلة GetColumnsInError، مع ملاحظة أن عليك فحص كلتيهما، لأنهما لا تحتويان على نفس البيانات!

**خطأ الصف RowError:**

تقرأ أو تغيّر النصّ الذي يصف الخطأ الذي حدث في هذا الصفّ.. لاحظ أن وضع أي نص في هذه الخاصية يغير قيمة الخاصية HasErrors إلى True، ويجعل جدول العرض يضع أيقونة الخطأ بجوار هذا الصف.

### حالة الصف RowState:

تعيد إحدى قيم المرقم DataRowState التي توضح حالة السجلّ من حيث كونه مضافا أو محذوفا أو معدّلا.. وقد تعرفنا على قيم هذا المرقم سابقا.

كما تمتلك هذه الفئة الوسائل التالية:

### له نسخة HasVersion:

تعيد True، إذا كان السجل الحالي يمتلك نسخة البيانات الموضحة في المعامل المرسل إلى هذه الوسيلة، وهو من نوع المرقم DataRowVersion الذي تعرفنا عليه من قبل.
والمثال التالي يعرض قيمة الخانة الأولى في نسخة السجل المقترحة إن وجدت:

```
If Row.HasVersion(DataRowVersion.Proposed) Then
   MsgBox(Row(0, DataRowVersion.Proposed))
End If
```

### تغيير الحالة إلى مضاف SetAdded:

تغير قيمة الخاصية RowState إلى Addded.. وتسبب هذه الوسيلة خطأ في البرنامج إذا كان كانت حالة السجل تشير إلى أنه معدل.. ولحل هذه المشكلة، عليك استدعاء الوسيلة AcceptChanges أولا.

### تغيير الحالة إلى معدل SetModified:

تغير قيمة الخاصية RowState إلى Modified.. وتسبب هذه الوسيلة خطأ في البرنامج إذا كان كانت حالة السجل تشير إلى أنه معدل.. ولحل هذه المشكلة، عليك استدعاء الوسيلة AcceptChanges أولا.

### قبول التغييرات AcceptChanges:

يؤدّي استدعاء هذه الوسيلة إلى قبول التغييرات التي حدثت على الصف الحالي منذ أن تمّ تحميله من قاعدة البيانات، أو منذ آخر مرّة تمّ فيها استدعاء الوسيلة AcceptChanges.. ليس معنى هذا أنّ هذه التغييرات سيتمّ حفظها في قاعدة البيانات، ولكن سيتمّ النظر إليها على أنها البيانات الأصليّة للسجل، ولن يمكنك التراجع عنها.. لاحظ أن هذه الوسيلة تفعل ما يلي:

- تضع القيمة Unchanged في الخاصية RowState للسجل إذا كانت قيمتها تشير إلى أنه معدل أو مضاف.

- تزيل السجل من الجدول نهائيا إذا كانت حالته تشير إلى أنه محذوف Deleted.
- تنقل القيم من النسخة الحالية Current Version إلى النسخة الأصلية Original Version للسجل.

وعليك أن تستخدم هذه الوسيلة بحذر، حتى لا تضيع التغييرات التي حدث للسجل الحالي دون حفظها في الجدول الأصلي في قاعدة البيانات.
وتسبب هذه الوسيلة خطأ في البرنامج، إذا حاولت استدعائها لقبول تغييرات صف ليس مضافا إلى أي جدول!

### رفض التغييرات RejectChanges:

يؤدّي استدعاء هذه الوسيلة إلى رفض كلّ التغييرات التي أجريت على السجل الحالي، بحيث يعود إلى الحالة التي كان عليها عند تحميله من قاعدة البيانات أو عند آخر استخدام للوسيلة AcceptChanges.. لاحظ أن هذه الوسيلة تفعل التالي:

- تضع القيمة Unchanged في الخاصية RowState للسجل إذا كانت حالته تشير إلى أنه معدل أو محذوف.
- تزيل السجل من الجدول نهائيا إذا كانت حالته تشير إلى أنه مضاف Added.
- تنقل القيم من النسخة الأصلية Original Version إلى النسخة الحالية Current Version للسجل.

وتتيح لك هذه الوسيلة استعادة القيم الأصلية للسجل.. هذا مفيد في بعض الحالات، مثل التخلص من القيم التي سببت خطأ في السجل.

### بدأ التحرير BeginEdit:

تبدأ عمليّة تحرير الصف الحالي، وتعطل أحداث الجدول التي تنطلق عند حدوث تغيرات في السجلات، كما توقف عمليّات التحقق من صحّة البيانات المدخلة في كل خانة من خانات السجل، مما يتيح للمستخدم تحرير كل خانات السجل بدون أي اعتراض.
ويتم استدعاء هذه الوسيلة تلقائيا عندما يحاول المستخدم تحرير بيانات السجل المعروض في أدوات ربط البيانات Data-bound Controls مثل مربعات النصوص وجدول عرض البيانات DataGridView.
لاحظ أن السجل يحتفظ بالبيانات التي يتم إدخالها أثناء عملية التحرير في النسخة المقترحة Proposed Version.

### إلغاء التحرير CancelEdit:

تلغي عمليّة التحرير التي بدأت باستدعاء الوسيلة Begin Edit، وتتخلص من نسخة السجل المقترحة Proposed Version، وتحتفظ بالنسخة الحالية Current Version كما هي.. هذا معناه إلغاء التغييرات التي حدثت على السجل أثناء عملية التحرير.

### إنهاء التحرير EndEdit:

تنهي عمليّة التحرير التي بدأت باستدعاء الوسيلة Begin Edit، وتفحص القيم التي تم إدخالها في السجل أثناء وضع التحرير، فإن كانت صحيحة تقوم بحفظ نسخة السجل المقترحة Proposed Version في النسخة الحالية Current Version.. هذا معناه حفظ التغييرات التي حدثت على السجل أثناء عملية التحرير.

ويتم استدعاء هذه الوسيلة تلقائيا عند استدعاء الوسيلة AcceptChanges.

### معرفة خطأ العمود GetColumnError:

تعيد نصّا يصف الخطأ الذي حدث في إحدى خانات السجلّ الحالي.. وتستقبل هذه الوسيلة معاملا يوضح العمود الذي توجد فيه هذه الخانة، سواء في صورة رقم العمود أو اسمه أو كائن العمود DataColumn الذي يمثله.

### تغيير خطأ العمود SetColumnError:

تسمح لك بوضع نصّ يصف الخطأ الذي حدث في إحدى خانات السجلّ الحالي.. ولهذه الوسيلة معاملان:

- المعامل الأول يوضح العمود الذي توجد فيه هذه الخانة، سواء في صورة رقم العمود أو اسمه أو كائن العمود DataColumn الذي يمثله.
- المعامل الثاني يستقبل النص الذي يشرح سبب الخطأ.

لاحظ أن هذه الوسيلة أكثر تفصيلا من الخاصية RowError، لأنها تحدد الخطأ الذي حدث في كل خانة على حدة.. وتؤدي الوسيلة SetColumnError إلى وضع القيمة True في الخاصية HasErrors، وإلى ظهور أيقونة الخطأ في جدول العرض في الخانة الناتجة من تقاطع الصف الحالي مع العمود الذي أرسلته كمعامل.. لكن هذه الوسيلة لا تؤثر على قيمة الخاصية RowError.

- **معرفة الأعمدة التي بها أخطاء GetColumnsInError:**

تعيد مصفوفة أعمدة DataColumn Array تحتوي على الأعمدة التي بها أخطاء في السجلّ الحالي.

- **محو الأخطاء ClearErrors:**

تمحو كلّ النصوص التي تشير إلى حدوث أخطاء في السجلّ.. هذا سيجعل الخاصية RowError والوسيلة GetColumnError تعيدان نصوصا فارغة.

- **حذف Delete:**

تضع القيمة Deleted في الخاصية RowState الخاصة بالسجل الحالي.. هذا يتيح لك التراجع عن حذف هذا السجل باستدعاء الوسيلة RejectChanges أو حذفه فعلا عند استدعاء الوسيلة AcceptChanges. لاحظ أن استخدام الوسيلة Delete مع سجل مضاف (RowState = Added) سيؤدي إلى حذف هذا السجل في الحال.

- **معرفة الصفوف التابعة GetChildRows:**

تعيد مصفوفة صفوف DataRow Array بها كلّ السجلات المرتبطة بعلاقة بهذا السجلّ في جداول أخرى.. ولهذه الوسيلة الصيغ التالية:

1. الصيغة الأولى تستقبل كائن العلاقة DataRelation الذي تريد استخدامه.. هذا ضروري، لأن السجل الحالي قد يكون له سجلات فرعية في أكثر من جدول، كما هو الحال في جدول الدول Countries، الذي له سجلات فرعية في جدولي المؤلفين والناشرين.
2. الصيغة الثانية تستقبل اسم العلاقة، لتبحث عنها في مجموعة العلاقات الفرعية ChildRelations الخاصة بالجدول الذي يوجد به السجل الحالي.
3. الصيغتان الثالثة والرابعة مماثلتان للصيغتين السابقتين، ولكنهما تزيدان بمعامل ثان من نوع المرقم DataRowVersion، ليمكنك من خلاله اختيار نسخة السجلات Version التي تريد قراءتها من الجدول الفرعي.

- **تغيير الصف الرئيسي SetParentRow:**

أرسل إلى هذه الوسيلة كائن السجلّ DataRow الذي تريد جعله السجل الرئيسي Master للسجل الحالي.. لاحظ أن السجل الرئيسي يمكن أن يكون

في جدول آخر (الجدول الرئيسي)، أو أن يكون في الجدول الحالي (علاقة ذاتية Self Relation).
وتوجد صيغة أخرى لهذه الوسيلة، تزيد على الصيغة السابقة بمعامل ثان، يستقبل كائن العلاقة DataRelation الذي يربط بين السجلين.
ويمكنك استخدام هذه الوسيلة إذا أردت تصحيح خطا في الجدول الفرعي، كأن تغير مؤلف أحد الكتب بعد نسبته خطأ إلى مؤلف آخر.. وفي هذه الحالة كل ما ستفعله هذه الوسيلة، هي وضع قيمة المفتاح الرئيسي ID للمؤلف، في خانة المفتاح الفرعي AuthorID للكتاب.

**معرفة الصف الرئيسي GetParentRow:**
تعيد السجلّ الرئيسي الذي يرتبط به السجلّ الحالي بعلاقة.. ولهذه الوسيلة عدة صيغ:

1. بعض الصيغ تستقبل معاملا واحدا، هو العلاقة التي يشترك فيها السجل الحالي، سواء في صورة اسم العلاقة، أو كائن العلاقة DataRelation.
2. بعض الصيغ تستقبل معاملا ثانيا من نوع المرقم DataRowVersion، ليمكنك من خلاله تحديد النسخة Version التي تريد قرأءتها من السجل الرئيسي.

**معرفة الصفوف الرئيسية GetParentRows:**
مماثلة للوسيلة السابقة في صيغها، لكنها تعيد مصفوفة صفوف DataRow Array، تحتوي على كلّ السجلات الرئيسيّة التي تشير إلى السجلّ الحالي.. في الحقيقة لا تبدو لهذه الوسيلة أية أهمية حاليا، فهي دائما تعيد سجلا رئيسيا واحدا، وهذا يجعل استخدام الوسيلة GetParentRow أكثر منطقية!

**هل هي عدم IsNull:**
تعيد True إذا كانت الخانة الموجودة في السجلّ الحالي والعمود المرسل كمعامل فارغة DbNull.. ولهذه الوسيلة الصيغ التالية:

1- بعض الصيغ لها معامل واحد، يستقبل اسم العمود أو رقمه أو كائن العمود DataCoulmn الذي يمثله.
2- وهناك صيغ لها معامل ثان من نوع المرقم DataRowVersion، ليمكنك من خلاله تحديد النسخة Version التي تريد فحص قيمها.

## فئة مجموعة الأعمدة DataColumnCollection Class

هذه الفئة ترث المجموعة InternalDataCollectionBase، وهي تحتوي على عناصر من نوع فئة عمود البيانات DataColumn Class.
ولا تملك هذه المجموعة أية خصائص أو وسائل جديدة غير ما ترثه من الفئة الأم، ولكن بعض هذه العناصر يحتاج منا إلى وقفة:

### العنصر Item:

تعيد كائن عمود البيانات DataColumn الموجود في المجموعة بناء على المعامل المرسل إليها.. ولهذه الوسيلة صيغتان:

1- الصيغة الأولى تستقبل رقم الخانة التي يوجد بها العمود في المجموعة.. لاحظ أن خطأ سيحدث لو أرسلت رقم خانة غير موجودة في المجموعة.

2- والصيغة الثانية تستقبل نصا يمثل اسم العمود في الجدول، ليتم البحث عنه في المجموعة، فإن كان موجودا تعيد هذه الوسيلة كائن العمود الذي يمثله، وإن لم يكن موجودا فإنها تعيد Nothing ولا يحدث خطأ.

### إضافة Add:

تضيف عمودا إلى مجموعة الأعمدة، مع ملاحظة أن الأعمدة التي تضيفها إلى مجموعة البيانات هي أعمدة مؤقتة خاصة بالبرنامج فقط، ولا تظهر في قاعدة البيانات، حتى بعد إجراء عملية التحديث Update.. لكن لو كنت تحتاج إلى إنشائها في قاعدة البيانات، فعليك باستخدام أوامر SQL الخاصة بإنشاء أعمدة تناظر الأعمدة الجديدة التي أضفتها في مجموعة البيانات.
وتمتلك هذه الوسيلة الصيغ التالية:

1- الصيغة الأولى بدون معاملات، وهي تنشئ عمودا جديدا بالاسم الافتراضي (Column1 أو Column2 ... إلخ).. لاحظ أن هذا العمود سيتعامل مع بيانات نصية String.

2- الصيغة الثانية تستقبل كائن العمود DataColumn وتضيفه إلى المجموعة.

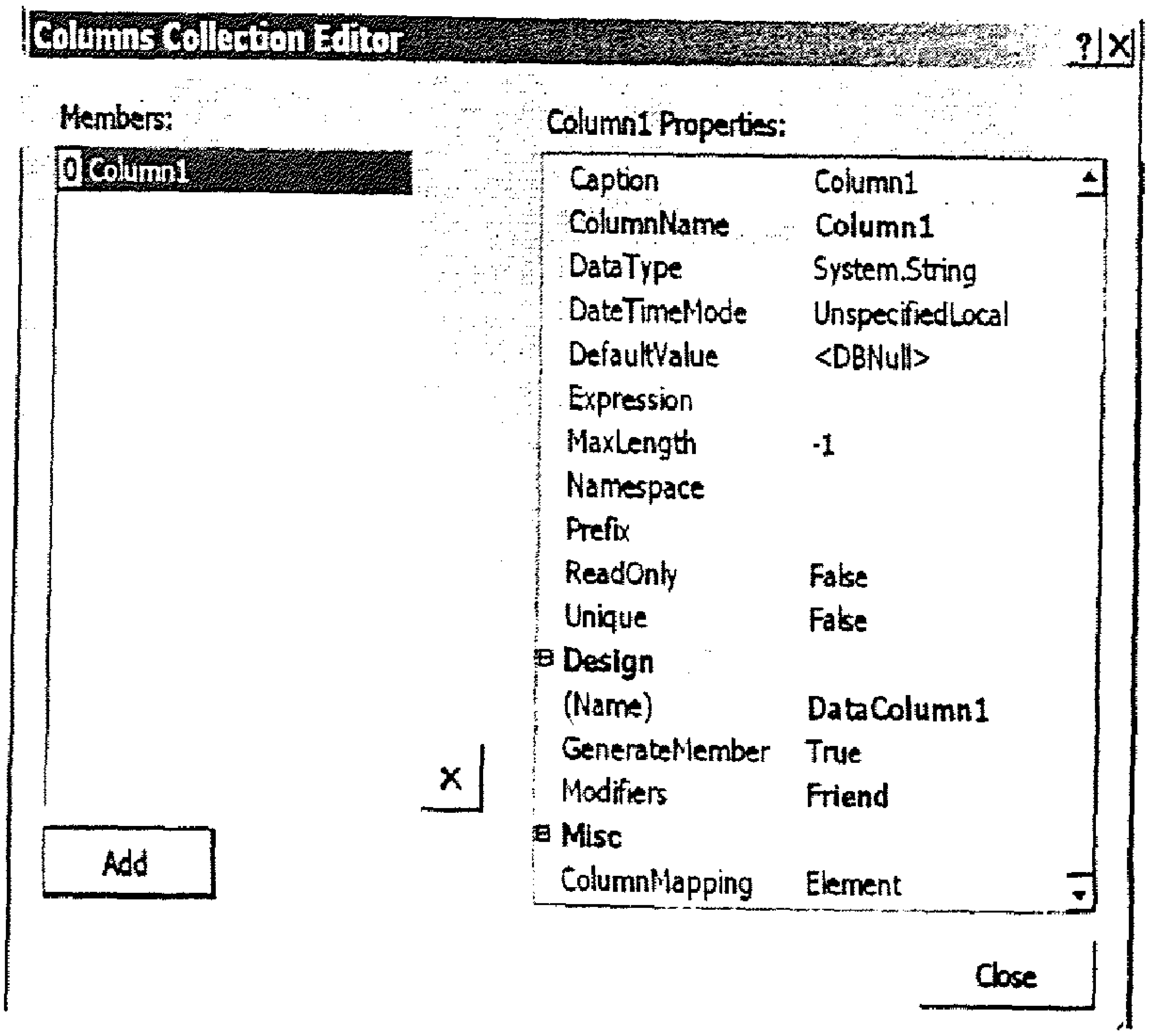
3- الصيغة الثالثة تستقبل اسم العمود، وتقوم بإنشائه وإضافته إلى المجموعة.. وتستطيع إرسال نص فارغ إلى هذه الوسيلة، لإنشاء عمود له الاسم الافتراضي (Column1 أو Column2 ... إلخ).

4- الصيغة الرابعة تزيد على الصيغة السابقة بمعامل ثان من نوع فئة النوع Type، ليمكنك من خلاله تحديد نوع بيانات العمود.. لاحظ أن نوع العمود يعتبر نصيا String في الصيغ التي لا تستقبل هذا المعامل.

5- الصيغة الخامسة تزيد على الصيغة السابقة بمعامل ثالث، يستقبل نصا يمثل الصيغة التي ستوضع في الخاصية Expression الخاصة بالعمود، مما يتيح لك إنشاء عمود محسوب Calculated Column.. وسنتعرف على هذه الخاصية عند التعرف على فئة عمود البيانات DataColumn.

لاحظ أن هذه الصيغ تعيد كائن العمود DataColumn الذي أضيف إلى مجموعة الأعمدة، ما عدا الصيغة الثانية فأنت ترسل إليها كائن العمود بالفعل لهذا ليست لها قيمة عائدة.

ويمكنك إضافة الأعمدة إلى هذه المجموعة بطريقة مرئية في وقت التصميم، وذلك من خلال نافذة خصائص الجدول.. لفعل هذا يجب أن يكون لديك كائن جدول في صينية مكونات النموذج (وهذا غير شائع)، أو يمكنك استخدام مجموعة بيانات عادية DataSet Un-Typed موضوعة في صينية المكونات، فلو عرضت خصائصها في نافذة الخصائص، فسيمكنك استخدام الخاصية Tables لعرض محرر مجموعة الجداول، ولو حددت أي جدول في هذه المجموعة، فستظهر خصائصه في القسم الأيمن من النافذة، وستجد بينها الخاصية Columns.. ولو ضغطت زر الانتقال الموجود في خانة هذه الخاصية، فستظهر نافذة محرر مجموعة الأعمدة، كما في الصورة:



في هذه النافذة يمكنك ضغط الزر Add لإضافة عمود جديد، حيث ستظهر خصائص هذا العمود في القسم الأيمن، ويمكنك تغييرها كما تشاء.

**موضع العمود IndexOf:**
تبحث عن العمود المرسل إليها كمعامل في مجموعة الأعمدة، وتعيد رقم الخانة التي يوجد بها في المجموعة، أو تعيد -1 إن لم يكن موجودا.. ولهذه الوسيلة صيغتان:

1- الصيغة الأولى تستقبل اسم العمود.
2- والصيغة الثانية تستقبل كائن العمود DataColumn.

**يمكن حذفه CanRemove:**
تعيد True إذا كان من الممكن حذف كائن العمود DataColumn المرسل كمعامل من مجموعة الأعمدة.. وتعيد هذه الوسيلة False إذا لم يكن العمود موجودا في المجموعة، أو إذا كان داخلا في علاقة.

**حذف Remove:**
تحذف العمود المرسل إليها كمعامل من مجموعة الأعمدة.. ولهذه الوسيلة صيغتان:

1- الصيغة الأولى تستقبل اسم العمود.
2- والصيغة الثانية تستقبل كائن العمود DataColumn.

كما تمتلك مجموعة الأعمدة الحدث التالي:

**المجموعة تغيرت CollectionChanged:**
ينطلق عندما يتغير عدد الأعمدة، سواء بالحذف أو الإضافة.
والمعامل الثاني e لهذا الحدث من النوع CollectionChangeEventArgs الذي تعرفنا عليه من قبل في مجموعة الجداول DataTableCollection.

## فئة عمود البيانات DataColumn Class

تمثل هذه الفئة مخطط أحد أعمدة الجدول، وهي ترث الفئة MarshalByValueComponent، مما يتيح لك إضافتها إلى صينية مكونات النموذج، وإن كان عليك أن تضيفها أولا إلى صندوق الأدوات.

ولحدث إنشاء هذه الفئة نفس صيغ الوسيلة Add الخاصة بمجموعة الأعمدة DataColumnCollection، ما عدا الصيغة الثانية.. كما توجد صيغة إضافية لحدث الإنشاء، تستقبل المعاملات التالية بالترتيب:

- اسم العمود.
- كائن نوع Type Object، يحتوي على نوع بيانات العمود.
- صيغة العمود التي ستوضع في الخاصية Expression التي سنتعرف عليها بعد قليل.
- إحدى قيم المرقم MappingType، لتوضع في الخاصية ColumnMapping الخاصة بالعمود، وسنتعرف عليها بعد قليل.

والمثال التالي يعرف عمودا نصيا اسمه Temp ويضيفه إلى جدول الكتب في مجموعة البيانات التي تحمل الاسم Ds:

```
Dim Clmn As New DataColumn("Temp", GetType(String))
Ds.Tables("Books").Columns.Add(Clmn)
```

وتمتلك فئة العمود الخصائص التالية:

### الجدول Table:

تعيد كائن الجدول DataTable الذي ينتمي إليه هذا العمود.

### اسم العمود ColumnName:

تقرأ أو تغير اسم العمود.

### نطاق الاسم Namespace:

تقرأ أو تغير نطاق الاسم الخاص بالجدول الذي ينتمي إليه العمود.

### البادئة Prefix:

تقرأ أو تغير البادئة التي تمثل نطاق اسم الجدول الذي ينتمي إليه العمود.

**الرتبة Ordinal:**

تعيد عددا صحيحا يمثل ترتيب العمود في مجموعة الأعمدة.

**العنوان Caption:**

تقرأ أو تغير عنوان العمود.. من المفروض أن يتم عرض هذا العنوان بدلا من اسم العمود في أدوات ربط البيانات Data-Bound Controls، لكنك لو جربت هذا مع جدول العرض مثلا، فلن تجد له تأثيرا!.. يبدو أن عليك ربط أداة العرض حصريا بهذه الخاصية، لكي ترى تأثيرها، كما سنرى لاحقا.

**نوع البيانات DataType:**

تقرأ أو تغير كائن النوع Type Object الذي يمثل نوع بيانات العمود.

**القيمة الافتراضية DefaultValue:**

تقرأ أو تغير القيمة الافتراضيّة لخانات العمود.. يمكنك مثلا أن تضع الرقم صفر في هذه الخاصية إذا كنت تتعامل مع عمودي رقمي.

**السماح بالعدم AllowDBNull:**

إذا جعلت قيمة هذه الخاصيّة True، فسيُسمح بترك بعض خانات هذا العمود فارغة DbNull.

**أقصى طول MaxLength:**

تقرأ أو تغير أقصى عدد من الحروف يمكنك كتابته في العمود الذي يتعامل مع بيانات نصية.. والقيمة الافتراضية لهذه الخاصية -1 مما يعني عدم وجود قيود على عدد الحروف.. لاحظ أن قيمة هذه الخاصية سيتم تجاهلها إذا كان العمود يتعامل مع بيانات من نوع آخر غير النصوص.

**للقراءة فقط ReadOnly:**

إذا جعلت قيمة هذه الخاصيّة True، فلن يمكنك تغيير قيمة أي خانة في هذا العمود بعد إضافة الصف الذي توجد به إلى الجدول.. لاحظ أن جعل قيمة هذه الخاصية True مع عمود يحمل ناتج عملية حسابية موضوعة في الخاصية Expression، سيؤدي إلى حدوث خطأ في

البرنامج، لأن تغيير قيمة أي خانة في عمود داخل في العملية الحسابية سيؤدي تلقائيا إلى إعادة حساب قيمة الخانة المناظرة في عمود الناتج.

**متفرد Unique:**
إذا جعلت قيمة هذه الخاصيّة True، فلن يسمح بتكرار قيم خانات هذا العمود.

**ترقيم تلقائي AutoIncrement:**
إذا جعلت قيمة هذه الخاصيّة True، فستزيد قيمة خانات هذا العمود تلقائيّا كلما تمت إضافة صف جديد إلى الجدول.. لاحظ أن عمود الترقيم التلقائي يكون للقراءة فقط ReadOnly، ولا يمكنك تغيير قيمته بنفسك.. والقيمة الافتراضية لهذه الخاصية هي False.

**بذرة الترقيم التلقائي AutoIncrementSeed:**
تحدد رقم بدء الترقيم في عمود الترقيم التلقائيّ.. والقيمة الافتراضية هي 1.

**خطوة الترقيم التلقائي AutoIncrementStep:**
تحدّد مقدار الزيادة في عمود الترقيم التلقائيّ.. والقيمة الافتراضية هي 1.

**الخصائص الإضافية ExtendedProperties:**
تعيد مجموعة الخصائص PropertyCollection التي تحتوي على الخصائص الإضافيّة للعمود، وهي مماثلة لتلك الخاصّة بكائن الجدول.

**خريطة العمود ColumnMapping:**
تحدد كيف يتم تمثيل العمود في كود XML عند حفظ الجدول، وهي تأخذ إحدى قيم المرقم MappingType التالية:

| | |
|---|---|
| Element | يتم تمثيل العمود كعنصر XML.. هكذا مثلا سيتم حفظ العمودين ID و Author في جدول المؤلفين:<br><Authors><br><ID>1</ID><br><Author> = 'توفيق الحكيم<br></Authors> |

| | |
|---|---|
| Attribute | يتم تمثيل العمود كسمة XML.. هكذا مثلا سيتم حفظ العمود ID:<br><Authors ID=1 Author = 'توفيق الحكيم'><br><Book ID=1 Book = 'شهرزاد'/><br></Authors> |
| SimpleContent | يتم تمثيل العمود كفرع من نوع الفئة XmlText.. هذا الموضوع خارج نطاق هذا الكتاب. |
| Hidden | هذا العمود خفي، ويستخدم في البناء الداخلي للجدول، لكن لا يظهر للمستخدم عند ربط الجدول بأدوات ربط البيانات. |

### نظام التاريخ والوقت DateTimeMode:

تحدد نظام التوقيت المستخدم مع هذا الجدول، إذا كان نوع بياناته تاريخ أو وقت.. وتأخذ هذه الخاصية إحدى قيم المرقم DataSetDateTime التالية:

| | |
|---|---|
| Local | التعامل بالتاريخ المحلي للجهاز الذي يوجد عليه البرنامج. |
| Utc | التعامل بالتاريخ العالمي. |
| Unspecified | غير محدد. |
| UnspecifiedLocal | توقيت محلي غير محدد.. هذه هي القيمة الافتراضية. |

لاحظ أنك لا تستطيع تغيير قيمة هذه الخاصية بعد إضافة الخانات إلى العمود، إلا في حالة واحدة: إذا كنت تحول من القيمة Unspecified إلى UnspecifiedLocal أو العكس.

### التعبير Expression:

تقرأ أو تغير الصيغة النصية للعمود، والتي يمكن استخدامها فيما يلي:

- حساب قيم خانات العمود الحالي، من ناتج عملية حسابية على أعمدة أخرى، وفي هذه الحالة يسمى بالعمود المحسوب Calculated Column.
- حساب ناتج شرط معين على كل سجل، لاستخدام الوسيلة DataTable.Select بعد ذلك للحصول على السجلات التي حققت هذا الشرط، وسنرى مثالا على هذا بعد قليل.

والجدول التالي يلخص لك المعاملات والدوال التي يمكنك استخدامها لتكوين صيغة العمود:

| | |
|---|---|
| المعاملات الحسابية | يمكنك استخدام المعاملات الحسابية التالية على الأعمدة:<br>الجمع: +، الطرح: -، الضرب *، القسمة /، باقي القسمة %.. مثال:<br>**Col.Expression = "Price * Copies_No"** |
| المعاملات المنطقية | يمنكن استخدام المعاملات المنطقية التالية بين الأعمدة:<br>AND, OR, NOT<br>والمثال التالي يرى إن كان اسم الكتاب يبدأ بحرف يسبق الميم أبجديا، وأن الكتاب للمؤلف رقم 4:<br>**Col.Expression =**<br>**"Book < 'م' And AuthorID = 4"**<br>لاحظ أن التعبير السابق سيضع في خانات العمود True أو False، مما يسمح لك باستخدام الوسيلة Select الخاصة بكائن الجدول لاختيار الصفوف التي تحقق أو لا تحقق شرطا معينا.. اعتبر أن العمود في المثال السابق اسمه Col.. هذا المثال يحصل على الصفوف التي تحقق الشرط السابق:<br>**Dim R( ) As DataRow**<br>**R = T.Select ("Col = True")** |
| معاملات المقارنة | يمكنك استخدام معاملات المقارنة التالية:<br>= <> > < >= <= IN LIKE<br>بنفس الطريقة التي تعرّفنا عليها عند شرح جمل SQL، مع ملاحظ أن علامات التعويض (* أو %) غير مسموح بها في منتصف التعبير المستخدم مع الدالة LIKE.. مثلا، التعبير التالي غير مقبول:<br>**"Author Like 'أحمد*توفيق' "**<br>لكن يمكن استخدام علامات التعويض في بداية النص أو نهايته.. مثال:<br>**"Author Like '*هيكل' "**<br>**"Author Like 'هيكل*' "** |

| | |
|---|---|
| CONVERT | تقوم بالتحويل بين أنواع البيانات.. مثال:<br>**Col.Expression =<br>"Convert(ID, 'System.Int32')"** |
| LEN | تعيد طول النصّ الموجود في خانات العمود.. والمثال التالي سيجعل العمود Col يعرض طول أسماء الكتب الموجودة في العمود Book:<br>**Col.Expression = "Len(Book)"**<br>لاحظ أنك غير مجبر على قصر معامل هذه الدالة (والدوال التالية أيضا) على اسم أحد حقول الجدول.. إنّ تعبيرا كهذا متاح أيضا:<br>**Col.Expression = "Len(Book + Author)"**<br>ولا تحاول استخدام العلامة "&" لتشبيك النصوص، فهي غير متاحة هنا.. استخدم بدلا منها علامة الجمع "+". |
| ISNULL | تستقبل هذه الدالة معاملين: اسم العمود، وقيمة افتراضية.. وتفحص هذه الدالة العمود، فإن وجدت فيه قيمة أعادتها، وإن وجدته فارغا أعادت القيمة الافتراضية.. المثال التالي يعيد قيمة العمود AuthorID أو −1 إذا كان فارغا:<br>**Col.Expression = "IsNull(AuthorID, -1)"** |
| TRIM | تزيل المسافات من بداية ونهاية خانات الحقل المرسل إليها كمعامل.. مثال:<br>**Col.Expression = "TRIM(AuthorID)"** |
| SUBSTRING | تعيد جزءا من النصّ الموجود في خانات العمود المرسل إليها كمعامل، بدءا من الحرف المذكور رقمه في المعامل الثاني، وبالطول المحدّد في المعامل الثالث.. (مماثلة للدالة String.SubString).. مثال:<br>**Col.Expression =<br>"SUBSTRING(Book, 2, 8)"** |
| IIF | تأخذ هذه الدالة ثلاثة معاملات:<br>− الأول شرط سيتمّ التحقق منه.<br>− والثاني هو القيمة المعادة إذا كان الشرط صحيحا.<br>− والثالث هو القيمة المعادة إذا كان الشرط خاطئا.<br>مثال:<br>**Col.Expression =<br>"IIF(LEN(Price)>10), 'غال', 'رخيص')"** |

| | |
|---|---|
| Parent | تشير إلى الجدول الرئيسي الذي يدخل في علاقة مع الجدول الحالي.. افترض أن المتغير Col يتعامل مع عمود في جدول الكتب.. المثال التالي يضع في هذا العمود طول اسم كل مؤلف:<br>**Col.Expression = "LEN(Parent.Author)"**<br>وإذا كان الجدول مشتركا في أكثر من علاقة مع جداول رئيسية اخرى، فيمكنك أن ترسل اسم العلاقة بين قوسين بعد الكلمة Parent كالتالي:<br>**Col.Expression =**<br>**"LEN(Parent(AuthorsBooks).Author)"** |
| Child | تشير إلى الجدول الثانويّ المرتبط بالجدول الحاليّ بعلاقة، حيث تستخدم المفتاح الأساسي Primary Key لكل سجل في الجدول الرئيسي، للحصول على السجلات الفرعية المرتبطة بهذا المفتاح في الجدول الفرعي.. ونظرا لأن ناتج هذه الدالة قد يكون أكثر من سجل، فلا يمكن استخدامها بمفردها، وإنما تستخدم مع إحدى دوال التجميع.<br>والمثال التالي يضيف عمودا إلى جدول المؤلفين، يعرض عدد الكتب التي ألفها كلّ مؤلف:<br>**Dim Col As New DataColumn(**<br>**"NoOfBooks", GetType(Integer))**<br>**Col.Expression = "Count(Child.AuthorID)"**<br>**Ds.Tables("Authors").Columns.Add(Col)**<br>ولو كان الجدول داخلا في أكثر من علاقة، فيمكنك إرسال اسم العلاقة كمعامل إلى التعبير Child، مثل:<br>**Col.Expression =**<br>**"Count(Child(AuthorsBooks).AuthorID)"** |
| دوال التجميع | يمكنك استخدام دوال التجميع التالية:<br>Sum, Avg, Min, Max, Count, StDev, Var<br>لاحظ أن استخدام هذه الدوال على العمود الحالي سيجعلها تتعامل مع كل الصفوف بدون تقسيمها إلى مجموعات Grouping.. مثلا، لو استخدمت التعبير: "Count (AuthorID)" في جدول الكتب فسيعيد عدد صفوف جدول الكتب.. ولو أردت تجميع الصفوف التي تتشابه في قيمة عمود معين، فعليك استخدام التعبير |

| | |
|---|---|
| | Child للاستفادة من العلاقة بين جدولين في تجميع الصفوف التي تتشابه في المفتاح الرئيسي.. ولو استخدمت التعبير: "Count (Child.AuthorID)" في جدول المؤلفين، فستحصل على مجموع كتب كل مؤلف. |
| \r | حرف نهاية السطر. |
| \n | حرف بداية السطر. |
| \t | علامة جدولة Tab. |

لاحظ أن عليك وضع النصوص والتواريخ بين العلامتين ' ' مثل:

'طويل'

'1/1/2009'

كما يمكن وضع التواريخ بين العلامتين # #، مثل:

#1/1/2009#

ويمتلك كائن العمود الوسيلة التالية:

- **تغيير الرتبة SetOrdinal:**

أرسل إلى هذه الوسيلة عددا صحيحا، يمثل الموضع الجديد الذي تريد أن يصير العمود فيه في مجموعة الأعمدة.. هذا هو الحل الوحيد لتغيير موضع العمود، لأن مجموعة الأعمدة لا تحتوي على الوسيلة Insert.. وقد استخدمنا هذه الوسيلة في الزر "حفظ البيانات 2" في المشروع CustomDataSet لجعل العمود Subject في الموضع رقم 2 (العمود الثالث) بعد إعادة إضافته إلى الجدول.. لاحظ أن هذا سيغير ترتيب العمود في الجدول، لكنه سيظهر في جدول العرض كآخر عمود!.. هذا لا يؤثر في عمل البرنامج، لكنه يزعج المستخدم، لهذا عليك تغيير ترتيب عرض العمود في جدول العرض أيضا.. هذا هو سبب استخدامنا للجملة التالية في نهاية الإجراء ShowGrades:

**GradesCols(2).DisplayIndex = 0**

هذه الجملة تجعلنا واثقين أن العمود الذي يعرض اسم المادة يظهر دائما قبل العمود الذي يعرض درجات الطالب.

## فئة قارئ جدول البيانات DataTableReader Class

هذه الفئة ترث فئة قارئ البيانات الأم DbDataReader Class، وهي تشبه قارئ البيانات العادي في طريقة عملها، لكنها لا تستخدم كائن أمر للحصول على السجلات من قاعدة البيانات، فهي تقرأ السجلات من جداول مجموعة البيانات مباشرة.

ولإنشاء قارئ بيانات يقرأ سجلات أحد الجداول، عليك باستخدام الوسيلة CreateDataReader الخاصة بهذا الجدول كالتالي:

```
Dim Tr = Ds.Tables("Authors").CreateDataReader
Do While Tr.Read
   MsgBox(Tr("ID"))
   MsgBox(Tr("Author"))
Loop
```

ولإنشاء قارئ بيانات يقرأ سجلات كل الجداول، عليك باستخدام الوسيلة CreateDataReader الخاصة بمجموعة البيانات DataSet كالتالي:

```
Dim Tr = Ds.CreateDataReader
Do
  Do While Tr.Read
     Dim RowTxt As String = ""
     For I = 0 To Tr.FieldCount - 1
        RowTxt &= Tr.GetName(I) + " = " +
                   Tr(I).ToString( ) + vbCrLf
     Next
     MsgBox(RowTxt)
  Loop
Loop While Tr.NextResult
```

لاحظ استخدامنا للوسيلة NextResult للانتقال من سجلات أحد الجداول إلى سجلات الجدول التالي.. لاحظ أيضا أننا لم نستخدم أسماء الأعمدة عند قراءة خانات كل سجل، وذلك لأن السجلات ستختلف من جدول إلى آخر في عدد الأعمدة وأسمائها.. وبدلا من هذا استخدمنا الخاصية FieldCount لإنشاء حلقة تكرار تمر عبر كل الأعمدة، لقراءة كل خانة باستخدام رقم العمود بدلا من اسمه.

ويمكنك تجربة هذا الكود في المشروع DataTableReaderSample.

ولحدث إنشاء الفئة DataTableReader الصيغتان التاليتين:

1- الأولى تستقبل كائن الجدول DataTable الذي ستقرأ سجلاته.

2- والثانية تستقبل مصفوفة جداول DataTable Array لتقرأ سجلاتها.

ولا تمتلك هذه الفئة أية خصائص أو وسائل جديدة غير ما ترثه من الفئة الأم.

## فئة مجموعة العلاقات
## DataRelationCollection Class

هذه الفئة ترث الفئة InternalDataCollectionBase، وهي تحتوي على كائنات من نوع فئة علاقة البيانات DataRelation Class.
وتمتلك هذه الفئة الخصائص والوسائل الشهيرة للمجموعات Collections، ومعظمها يستخدم اسم العلاقة كمعامل، أو يستخدم كائن العلاقة DataRelation الذي يمثلها.. لهذا نحتاج هنا إلى التركيز على العناصر التالية فقط:

### العنصر Item:
هذه هي الخاصية الافتراضية، وهي تستقبل اسم العلاقة كمعامل أو رقم العلاقة في المجموعة، وتعيد إليك كائن العلاقة DataRelation الذي يمثلها.. وقد استخدمنا هذه الخاصية في المشروع DataSetContents، للحصول على كائن العلاقة التي يضغط المستخدم اسمها في قائمة العلاقات، لنعرض خصائصها في مربع رسالة.

### إضافة Add:
تضيف علاقة إلى المجموعة، ولها الصيغ التالية:

1- الصيغة الأولى تستقبل كائن العلاقة DataRelation المراد إضافته.
2- الصيغة الثانية تستقبل معاملين من النوع DataColumn، يمثلان الحقل الرئيسي والحقل الفرعي على الترتيب، حيث سيتم إنشاء علاقة بينهما، وإضافتها إلى المجموعة.
3- الصيغة الثالثة مماثلة للصيغة السابقة، إلا أنها تستقبل مصفوفتين من النوع DataColumn، وذلك لمراعاة الحالة التي يتكون فيها كل من المفتاح الأساسي والفرعي من أكثر من عمود.
4- الصيغة الرابعة تستقبل ثلاثة معاملات: اسم العلاقة، وكائن العمود الرئيسي، وكائن العمود الفرعي.

5- الصيغة الخامسة تزيد بمعامل منطقي على الصيغة السابقة، إذا جعلت قيمته False فلن يتم إنشاء قيود عند إنشاء العلاقة.. والقيمة الافتراضية في الصيغ التي لا تحتوي هذا المعامل هي True، لهذا يتم إنشاء قيد التفرد UniqueConstraint على الحقل الأساسي إن لم يكن موجودا، وإنشاء قيد المفتاح الفرعي ForeignKeyConstraint على الحقل الفرعي إن لم يكن موجودا، ويتم إضافتهما إلى قيود الجدول.

6- الصيغة السادسة مماثلة للصيغة السابقة، إلا أن معامليها الثاني والثالث يستقبلان مصفوفة حقول DataColumn Array لمراعاة الحالة التي يتكون فيها كل من المفتاحين الأساسي والفرعي من أكثر من حقل.

لاحظ أن جميع الصيغ ـ ما عدا الأولى ـ تعيد كائن العلاقة DataRelation الذي تم إنشاؤه وإضافته إلى المجموعة.

### المجموعة تغيرت CollectionChanged:

ينطلق هذا الحدث عندما يتغير عدد عناصر مجموعة العلاقات، سواء بالحذف أو الإضافة.. والمعامل الثاني e لهذا الحدث من النوع CollectionChangeEventArgs الذي تعرفنا عليه من قبل في مجموعة الجداول DataTableCollection.

## فئة العلاقة DataRelation Class

تحتوي هذه الفئة على تفاصيل العلاقة المنشأة بين جدولين.
ولحدث إنشاء هذه الفئة نفس صيغ الوسيلة DataRelationCollection.Add ما عدا الصيغة الأولى التي تستقبل كائن علاقة.. وإضافة إلى هذه الصيغ، يمتلك حدث الإنشاء الصيغتين الجديدتين التاليتين:

1- الصيغة الأولى تستقبل المعاملات التالية:

- نصا يمثل اسم العلاقة.
- نصا يمثل اسم الجدول الرئيسي.
- نصا يمثل اسم الجدول الفرعي.
- مصفوفة نصية تحتوي على أسماء أعمدة المفتاح الأساسي.
- مصفوفة نصية تحتوي على أسماء أعمدة المفتاح الفرعي.
- معاملا منطقيا Boolean، لتوضع قيمته في الخاصية Nested الخاصة بكائن العلاقة.

2- الصيغة الثانية تستقبل المعاملات التالية:

- نصا يمثل اسم العلاقة.
- نصا يمثل اسم الجدول الرئيسي.
- نصا يمثل نطاق اسم Namespace الجدول الرئيسي.
- نصا يمثل اسم الجدول الفرعي.
- نصا يمثل نطاق اسم الجدول الفرعي.
- مصفوفة نصية تحتوي على أسماء أعمدة المفتاح الأساسي.
- مصفوفة نصية تحتوي على أسماء أعمدة المفتاح الفرعي.
- معاملا منطقيا Boolean، لتوضع قيمته في الخاصية Nested الخاصة بكائن العلاقة.

والمثال التالي يريك كيف تنشئ علاقة بين الحقل ID في جدول المؤلفين، والحقل AuthorID في جدول الكتب:

```
Dim ID As DataColumn = _
        Ds.Tables("Authors").Columns("ID")
Dim AuthorID As DataColumn = _
        Ds.Tables("Books").Columns("AuthorID")
Dim R As New DataRelation("AuthorsBooks", ID, AuthorID)
```

لاحظ أنّ هذه العلاقة لم توضع في مجموعة البيانات DataSet إلى الآن، لهـذا عليك إضافتها إلى مجموعة العلاقات بنفسك كالتالي:

**Ds.Relations.Add(R)**

وبعد تنفيذ الجملة الأخيرة، ستضاف هذه العلاقة تلقائيا إلى مجموعـة العلاقـات الرئيسية ParentRelations لجدول المؤلفين، ومجموعـة العلاقـات الفرعيـة ChildRelations لجدول الكتب.

وتمتلك فئة العلاقة الخصائص التالية:

**مجموعة البيانات DataSet:**

تعيد كائن مجموعة البيانات DataSet، الذي تنتمي إليه هذه العلاقة.

**اسم العلاقة RelationName:**

تقرأ أو تغيّر اسم العَلاقة.

**الجدول الرئيسي ParentTable:**

تعيد كائن جدول DataTable يمثل الجدول الرئيسيّ في هذه العلاقة.

**الجدول الفرعي ChildTable:**

تعيد كائن جدول DataTable يمثل الجدول الثانويّ في هذه العلاقة.

**الأعمدة الرئيسية ParentColumns:**

تعيد مصفوفة من النوع DataColumn تحتوي على الأعمدة الرئيسيّة فـي هذه العلاقة.

**الأعمدة الفرعية ChildColumns:**

تعيد مصفوفة من النوع DataColumn تحتوي على الأعمدة الثانويّة فـي هذه العلاقة.

**قيد المفتاح الرئيسي ParentKeyConstraint:**

تعيد كائن قيد التفـرد UniqueConstraint، المفـروض علـى المفتـاح الرئيسيّ في هذه العلاقة.

### قيد المفتاح الفرعي ChildKeyConstraint:

تعيد كائن قيد المفتاح الفرعي ForeignKeyConstraint، المفروض على المفتاح الثانويّ في هذه العلاقة.

### متداخلة Nested:

تفيد عند حفظ سجلات الجدول الرئيسي في ملف XML باستخدام الوسيلة WriteXml، فلو جعلت قيمة هذه الخاصية True، فسيتم حفظ السجلات الفرعية مع السجل الأصلي الذي تربطها به العلاقة الحالية.

### الخصائص الإضافية ExtendedProperties:

تعيد مجموعة الخصائص PropertyCollection التي تحتوي على الخصائص الإضافيّة للعمود، وهي مماثلة لتلك الخاصّة بكائن الجدول.

## فئة مجموعة القيود ConstraintCollection Class

هذه الفئة ترث المجموعة InternalDataCollectionBase، وكل عنصر من عناصرها من نوع فئة القيد Constraint Class التي سنتعرف عليها لاحقا.
وتمتلك هذه الفئة الخصائص والوسائل الشهيرة للمجموعات Collections، ومعظمها يستخدم اسم القيد كمعامل، أو يستخدم كائن القيد Constraint الذي يمثله.. لهذا نحتاج هنا إلى التركيز على العناصر التالية فقط:

### العنصر Item:

أرسل إلى هذه الخاصية رقم القيد في المجموعة، أو نصا يحمل اسم القيد، لتعيد إليك كائن القيد Constraint الذي يمثله.

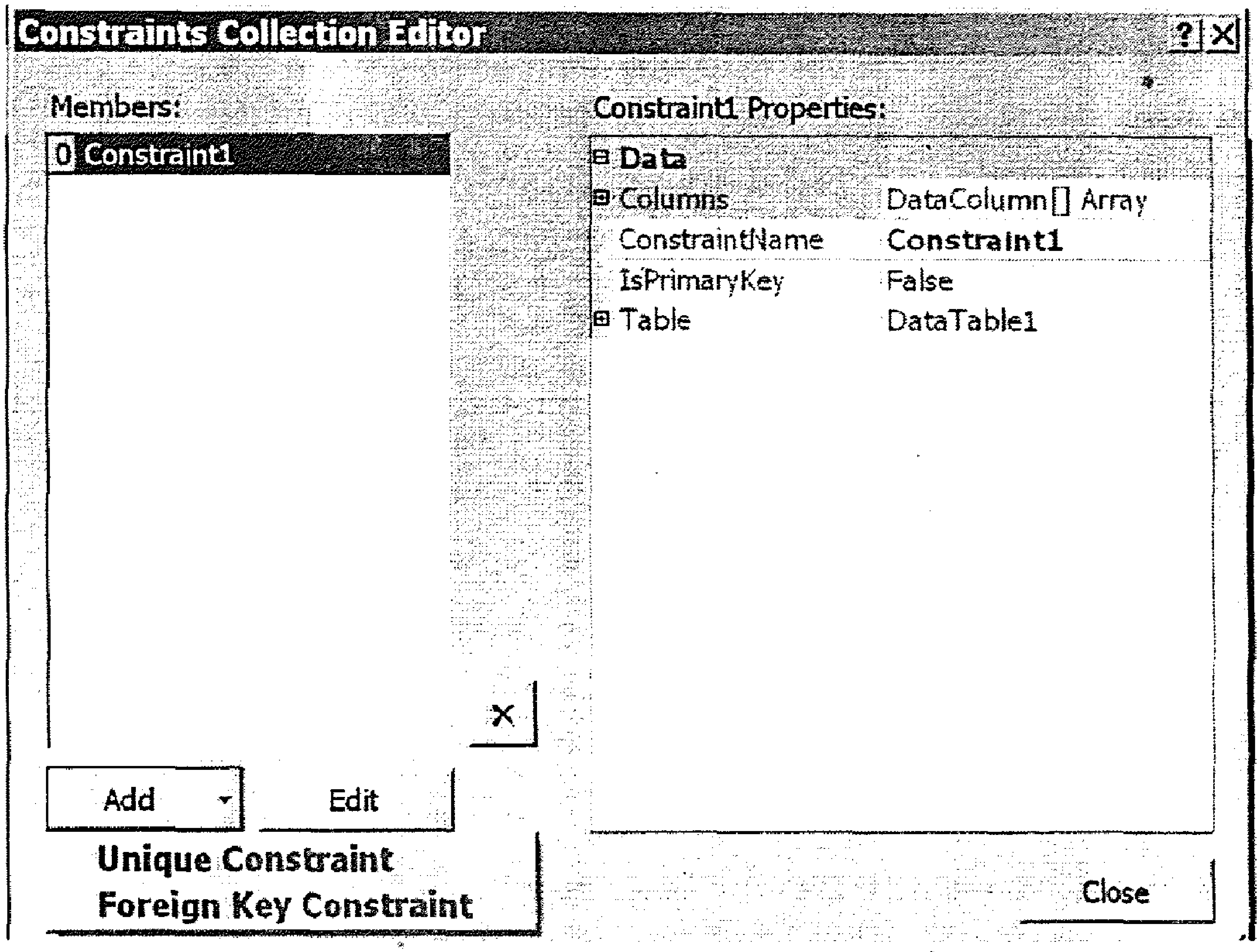
### إضافة Add:

تضيف قيدا إلى المجموعة، ولها الصيغ التالية:

1- الصيغة الأولى تستقبل كائن القيد Constraint المراد إضافته.
2- الصيغة الثانية تنشئ قيد تفرد UniqueConstraint وتضيفه إلى المجموعة.. وهي تستقبل ثلاثة معاملات:
   - اسم القيد.
   - كائن العمود DataColumn الذي يجب أن يكون متفردا.
   - قيمة منطقية إذا جعلتها True فسيتم جعل العمود المرسل إلى المعامل الثاني مفتاحا أساسيا للجدول.
3- الصيغة الثالثة مماثلة للصيغة السابقة، إلا أن معاملها الثاني يستقبل مصفوفة أعمدة DataColumn Array، وذلك إذا كان المفتاح المطلوب تفرده في الجدول يتكون من أكثر من عمود.
4- الصيغة الرابعة تنشئ قيد مفتاح فرعي ForeignKeyConstraint وتضيفه إلى المجموعة.. وهي تستقبل المعاملات التالية:
   - اسم القيد.
   - كائن العمود DataColumn الأساسي.
   - كائن العمود DataColumn الفرعي.
5- الصيغة الخامسة مماثلة للصيغة السابقة، إلا أن معامليها الثاني والثالث يستقبلان مصفوفة أعمدة DataColumn Array، وذلك إذا كان المفتاح الأساسي والمفتاح الفرعي يتكونان من أكثر من عمود.

لاحظ أن جميع الصيغ ما عدا الأولى، تعيد كائن القيد Constraint الذي تم إنشاؤه وإضافته إلى المجموعة.

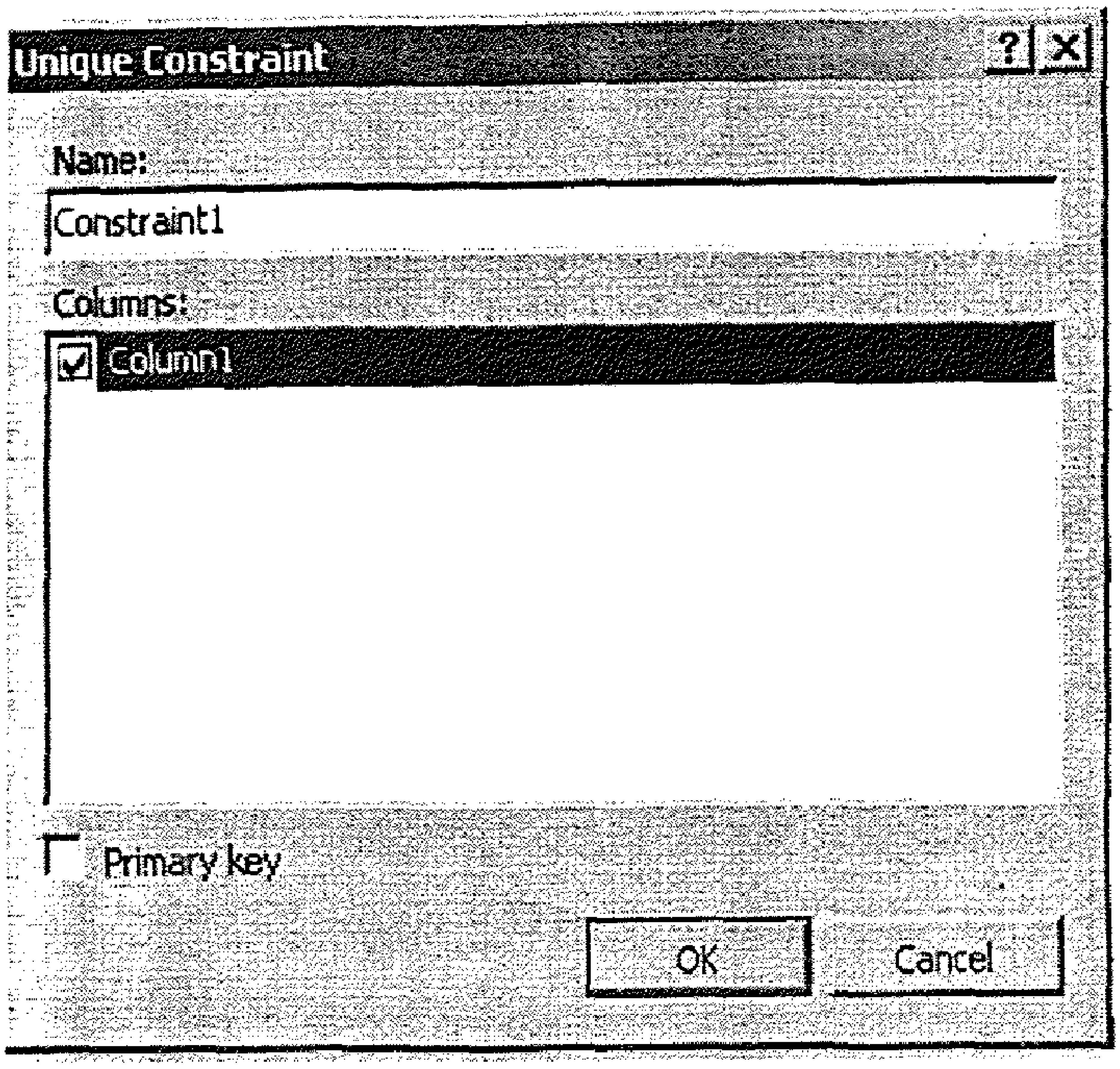
ويمكنك إضافة القيود إلى هذه المجموعة بطريقة مرئية في وقت التصميم، وذلك من خلال نافذة خصائص الجدول.. لفعل هذا يجب أن يكون لديك كائن جدول في صينية مكونات النموذج (وهذا غير شائع)، أو يمكنك استخدام مجموعة بيانات عادية DataSet Un-Typed موضوعة في صينية المكونات، فعند عرض خصائصها في نافذة الخصائص، سيمكنك استخدام الخاصية Tables لعرض محرر مجموعة الجداول، ولو حددت أي جدول في هذه المجموعة، فستظهر خصائصه في القسم الأيمن من النافذة، وستجد بينها الخاصية Constraints.. ولو ضغطت زر الانتقال الموجود في خانة هذه الخاصية، فستظهر نافذة محرر مجموعة القيود، كما في الصورة:




اضغط الزر Add لإضافة قيد جديد.. ستظهر لك قائمة موضعية لتتيح لك اختيار نوع القيد، من بين النوعين التاليين:

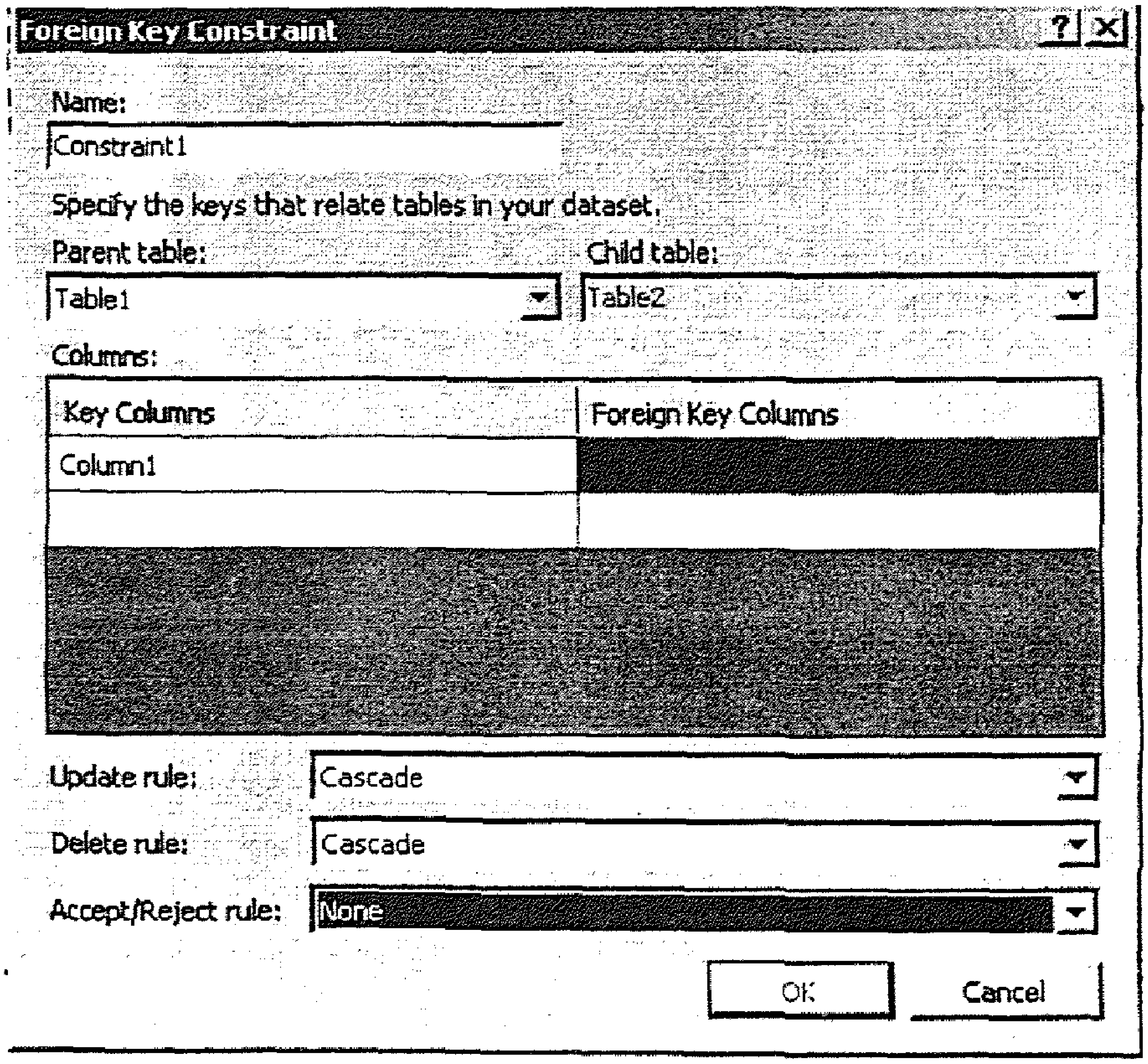
1- قيد التفرد Unique Constraint:

عند ضغط هذا الاختيار، ستظهر لك نافذة إنشاء قيد التفرد، وقد تعرفنا عليها في مخطط مجموعة البيانات، وهي تبدو كما في الصورة:




2- قيد المفتاح الفرعي Foreign Key Constraint:

عند ضغط هذا الاختيار، ستظهر لك نافذة إنشاء قيد المفتاح الفرعـــي، وهي تشبه نافذة إنشاء علاقة، ولا جديد فيها، وتبدو كما في الصـورة:



وبعد أن تضيف القيد سيظهر في القائمة اليسرى، وستظهر خصائصه في القائمة اليمنى.

**يمكن حذفه CanRemove:**
تعيد True إذا كان من الممكن أن تحذف من المجموعة، كائن القيد Constraint المرسل إليها كمعامل، بدون حدوث خطأ في البرنامج.. مثلا: محاولة حذف قيد التفرد UniqueConstraint قبل حذف قيد المفتاح الفرعي ForeignKeyConstraint المرتبط به، تؤدي إلى حدوث خطأ في البرنامج، لهذا عليك استخدام هذه الوسيلة قبل استخدام الوسيلة Remove أو RemoveAt.

**المجموعة تغيرت CollectionChanged:**
ينطلق هذا الحدث عندما يتغير عدد عناصر مجموعة القيود، سواء بالحذف أو الإضافة.. والمعامل الثاني e لهذا الحدث من النوع CollectionChangeEventArgs الذي تعرفنا عليه من قبل في مجموعة الجداول DataTableCollection.

## فئة القيد Constraint Class

هذه الفئة أساسة مجردة Abstract Base Class وتجب وراثتها، وهي تعمل كفئة أم لكل من فئة قيد التفرد UniqueConstraint Class وقيد المفتاح الفرعي ForeignKeyConstraint Class.
وتمتلك هذه الفئة الخصائص التالية:

**اسم القيد ConstraintName:**
تقرأ أو تغير اسم القيد.

**الجدول Table:**
تعيد كائن الجدول DataTable الذي ينطبق عليه القيد.

**الخصائص الإضافية ExtendedProperties:**
تعيد مجموعة الخصائص PropertyCollection التي تحتوي على الخصائص الإضافيّة للقيد، وهي مماثلة لتلك الخاصّة بكائن الجدول.

## فئة قيد التفرّد UniqueConstraint Class

هذه الفئة ترث الفئة Constraint، وهي تحتوي على تفاصيل قيد التفـرد الـذي يضمن عدم تكرار قيم حقل أو مجموعة من الحقول.
ولحدث إنشاء هذه الفئة الصيغ التالية:

1- الأولى تستقبل كائن العمود DataColumn الذي سيفرض عليه القيد.
2- الصيغة الثانية تزيد على الصيغة السابقة بمعامل منطقي، إذا جعلت قيمته True يتم جعل العمود المرسل إلى المعامل الأول مفتاحا أساسيا للجدول.
3- الصيغة الثالثة تستقبل مصفوفة أعمدة DataColumn Array تحتـوي على الأعمدة التي سيفرض عليها القيد.
4- الصيغة الرابعة تزيد على الصيغة السابقة بمعامل منطقي، إذا جعلته True فسيتم جعل الأعمدة المرسلة إلى المعامل الأول مفتاحا أساسيا للجدول.
5- هناك صيغ مماثلة للصيغ السابقة، لكنها تزيد بمعامل أول يستقبل اسم القيد.
6- الصيغة الأخيرة تستقبل ثلاثة معاملات:
   - اسم القيد.
   - مصفوفة نصية تستقبل أسماء الأعمدة.
   - معامل منطقي إذا جعلت قيمته True يتم جعل الأعمدة المرسلة إلـى المعامل الثاني مفتاحا أساسيا للجدول.

والكود التالي يعرف قيد التفرد على الحقل ID لجدول المؤلفين:

```
Dim ID As DataColumn = Ds.Tables(
    "Authors").Columns("ID")
Dim Uc As New UniqueConstraint("IDUnique", ID, True)
```

إلى الآن لم يوضع هذا القيد في جدول المؤلفين، لذا عليك أن تضيفه كالتالي:

```
Ds.Tables("Authors").Constraints.Add(Uc)
```

لكـن هـذا القيـد لـن يعمـل، إلا إذا جعلـت للخاصـيّة "فـرض القيـود" EnforceConstraints الخاصّة بمجموعة البيانات القيمة True:

```
Ds.EnforceConstraints = True
```

وإضافة إلى ما ترثه من الفئة الأم، تمتلك فئة قيد التفرد الخاصيتين التاليتين:

### الأعمدة Columns:

تعيد مصفوفة تحتوي على الأعمدة التي يؤثر عليها هذا القيد.

### هل هو مفتاح أساسي IsPrimaryKey:

تعيد True إذا كان هذا القيد مفروضا على المفتاح الأساسيّ للجدول.

## فئة قيد المفتاح الثانوي ForeignKeyConstraint Class

هذه الفئة ترث الفئة Constraint، وهي تحتوي على تفاصيل قيد المفتاح الثانوي، المفروض على السجلات في الجدول الثانويّ، والذي يضمن صحة العلاقة بين الجدولين، ويمنع حذف أحد السجلات الأساسية في الجدول الرئيسيّ إذا كانت له سجلات فرعية في الجدول الثانوي، كما يمنع تعديل قيمة المفتاح الأساسي في أي سجل إذا كانت هذه القيمة مستخدمة في المفتاح الفرعي لسجلات الجدول الثانوي.

ولحدث إنشاء هذه الفئة الصيغ التالية:

1- الصيغة الأولى تستقبل كائن العمود DataColumn الذي يمثل المفتاح الرئيسي في العلاقة، وكائن العمود DataColumn الذي يمثل المفتاح الفرعي في العلاقة، لفرض القيد عليهما.

2- الصيغة الثانية مماثلة للصيغة السابقة، إلا أنها تتعامل مع مصفوفتين من النوع DataColumn، لمراعاة الحالة التي يتكون فيها المفتاح الأساسي والمفتاح الفرعي من أكثر من عمود.

3- الصيغة الثالثة تزيد على كل من الصيغتين السابقتين بمعامل أول، يستقبل اسم القيد.

4- الصيغة الرابعة تستقبل المعاملات التالية بالترتيب:

- نصا يمثل اسم القيد.
- نصا يمثل اسم الجدول الرئيسي.
- نصا يمثل نطاق اسم الجدول.
- مصفوفة نصية تحتوي على أسماء الأعمدة التي تعمل كمفتاح أساسي.
- مصفوفة نصية تحتوي على أسماء الأعمدة التي تعمل كمفتاح فرعي.
- إحدى قيم المرقم AcceptRejectRule لوضعها في الخاصية AcceptRejectRule التي سنتعرف عليها بعد قليل.
- إحدى قيم المرقم Rule لوضعها في الخاصية DeleteRule التي سنتعرف عليها بعد قليل.
- إحدى قيم المرقم Rule لوضعها في الخاصية UpdateRule التي سنتعرف عليها بعد قليل.

5- الصيغة الخامسة مماثلة للصيغة السابقة، لكن ينقصها المعامل الثالث الذي يستقبل نطاق اسم الجدول.

والمثال التالي يعرّف قيدا من هذا النوع:

```
Dim ID As DataColumn = Ds.Tables(
    "Authors").Columns("ID")
Dim AuthorID As DataColumn = Ds.Tables(
    "Books").Columns("AuthorID")
Dim Fkc As New ForeignKeyConstraint("AuthorIDCnst",
    ID, AuthorID)
```

ويجب أن تضيف هذا القيد إلى الجدول الثانوي، مع ملاحظة أن خطأ سيحدث لو حاولت إضافته إلى الجدول الرئيسي.. والكود التالي يضيف القيد الذي أنشأناه إلى مجموعة قيود جدول الكتب:

```
Ds.Tables("Books").Constraints.Add(Fkc)
```

لاحظ أن تنفيذ هذه الجملة سيؤدي إلى إضافة قيد التفرد تلقائيا على العمود ID في جدول المؤلفين، وذلك إذا لم يكن هذا القيد موجودا مسبقا.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

### الجدول المرتبط RelatedTable:

تعيد كائن جدول DataTable يمثل الجدول الرئيسيّ في العلاقة.

### الأعمدة المرتبطة RelatedColumns:

تعيد مصفوفة أعمدة DataColumn Array، تحتوي على الأعمدة التي تعمل كمفتاح أساسي في هذه العلاقة.

### الأعمدة Columns:

تعيد مصفوفة أعمدة DataColumn Array، تحتوي على الأعمدة التي تعمل كمفتاح ثانوي في هذه العلاقة.

### قاعدة القبول والرفض AcceptRejectRule:

توضّح الفعل الذي سيتمّ اتخاذه عند استدعاء الوسيلة DataRow.AcceptChanges أو الوسيلة DataRow.RejectChanges.. وتأخذ هذه الخاصيّة إحدى قيمتي المرقم AcceptRejectRule التاليتين:

| | |
|---|---|
| None | عند قيام أحد السجلات بقبول التغييرات أو رفضها، لا يحدث أي شيء للسجلات المرتبطة به في العلاقة.. هذه هي القيمة الافتراضية. |
| Cascade | عند قيام أحد السجلات بقبول التغييرات أو رفضها، يتم قبول التغييرات أو رفضها في السجلات المرتبطة به في العلاقة.. مثلا: لو تم قبول تغييرات سجل في جدول المؤلفين، يتم قبول التغييرات على التوالي لكل السجلات التي تحتوي على كتب هذا المؤلف في جدول الكتب. |

### قاعدة الحذف DeleteRule:

توضّح الفعل الذي سيتمّ اتخاذه مع السجلات الفرعية، عند حذف السجل الرئيسي الذي تنتمي إليه، وهي تأخذ إحدى قيم المرقم Rule التالية:

| | |
|---|---|
| None | تترك السجلات الفرعية كما هي بدون تغيير، لكن هذا قد يؤدي إلى حدوث خطأ في البرنامج، لأن حذف أو تعديل السجل الرئيسي سيكسر قيد المفتاح الثانوي، ويترك بعض السجلات في الجدول الثانوي تشير إلى سجل غير موجود في الجدول الرئيسي. |
| Cascade | يؤدّي حذف أو تعديل السجلّ الرئيسي إلى حذف أو تعديل كلّ السجلات الثانويّة المرتبطة به.. فمثلا، سيؤدّي حذف أحد المؤلفين من جدول المؤلفين، إلى حذف كلّ كتبه من جدول الكتب. |
| SetDefault | يؤدّي حذف أو تعديل السجلّ الرئيسي إلى وضع القيم الافتراضيّة في حقول المفتاح الثانوي المرتبطة به. |
| SetNull | يؤدّي حذف أو تعديل السجلّ الرئيسي إلى إفراغ حقول المفتاح الثانوي المرتبطة لتصير بها القيمة DbNull. |

### قاعدة التحديث UpdateRule:

توضّح الفعل الذي يتمّ اتخاذه مع السجلات الفرعية، عند تغيير قيم السجل الرئيسي الذي تنتمي إليه، وهي تأخذ إحدى قيم المرقم Rule التي تعرفنا عليها الخاصية السابقة.

# -14-

# عـروض البيـانـات

# Data Views

تتيح لك العروض View عرض بعض أو كل سجلات الجدول بالترتيب الـذي تريده، دون التأثير على سجلات الجدول الأصلي.. هذا يمنحك مرونة عالية عند عرض البيانات للمستخدم، دون الحاجة إلى إعادة إرسال استعلامات مختلفة إلـى قاعدة البيانات.

وفكرة العرض بسيطة، فكائن العرض يحتوي على فهرس Index يـشير إلـى سجلات الجدول، ولا يحتوي على السجلات نفسها.. هذا يجعل كـائن العـرض سريعا في أداء عمليـات الترشـيح Filtering والترتيـب Sorting والبحـث Searching، دون أن يستهلك مساحة كبيرة في الذاكرة!

لاحظ أن فهرس السجلات يتم إنشاؤه عند إنشاء كائن العرض، ويعاد إنشاؤه مرة أخرى إذا تم تغيير طريقة الترتيب أو الترشيح.. لـذا مـن الأفـضل أن تحـدد مواصفات الترتيب والترشيح عند إنشاء كائن العرض لتوفر على نفـسك الوقـت الضائع في إعادة إنشاء الفهرس.

وفي هذا الفصل سنتعرف على الفئات التي تتيح لنا إنشاء العـروض والتعامـل معها.

# واجهة قائمة الربط
# IBindingList Interface

هذه الواجهة ترث واجهة القائمة IList، وهي تقدم الوسائل اللازمة للتعامل مع مصدر البيانات Data Source من خلال أدوات ربط البيانات Data-Bound Controls.

وتمتلك هذه الواجهة الخصائص التالية:

**السماح بالتحرير AllowEdit:**

تعيد True إذا كان من الممكن تغيير قيمة أي عنصر في القائمة.

**السماح بالجديد AllowNew:**

تعيد True إذا كان من الممكن إضافة عنصر جديد إلى القائمة باستخدام الوسيلة AddNew.

**السماح بالحذف AllowRemove:**

تعيد True إذا كان من الممكن حذف عنصر من القائمة باستخدام الوسيلتين Remove و RemoveAt.

**تدعم الترتيب SupportsSorting:**

تعيد True إذا كانت القائمة تسمح بترتيب عناصرها.. وإذا كانت قيمة هذه الخاصية False، فستؤدي محاولة استخدام أي من خصائص الترتيب إلى حدوث خطأ من النوع NotSupportedException.

**تدعم البحث SupportsSearching:**

تعيد True إذا كانت القائمة تتيح البحث في عناصرها، وتتيح ترتيبها.

**هل هي مرتبة IsSorted:**

تعيد True إذا كانت عناصر القائمة مرتبة.

**اتجاه الترتيب SortDirection:**
توضح اتجاه ترتيب القائمة، وهي تعيد إحدى قيمتي المرقم ListSortDirection التاليتين:

| | |
|---|---|
| Ascending | ترتيب تصاعدي. |
| Descending | ترتيب تنازلي. |

**خاصية الترتيب SortProperty:**
تعيد كائنا من نوع فئة واصف الخاصية PropertyDescriptor Class، وهو يحتوي المعلومات اللازمة لمعرفة الخاصية المستخدمة في ترتيب عناصر القائمة، وذلك إذا كانت القائمة تحتوي على كائنات Objects تمتلك خصائص مختلفة.. مثلا عند التعامل مع كائن عرض فيه سجلات أحد الجداول، يمكنك أن تحدد أحد القول لترتيب السجلات تبعا له.

**تدعم التنبيه عن التغيير SupportsChangeNotification:**
تعيد True إذا كانت القائمة تطلق الحدث ListChanged عند حدوث تغير في عناصرها.

كما تمتلك هذه الواجهة الوسائل التالية:

- **إضافة فهرس AddIndex:**
أرسل إلى هذه الخاصية كائنا من النوع PropertyDescriptor، لتقوم بإضافة الخاصية التي يشير إليها، إلى مجموعة الفهارس المستخدمة في البحث في عناصر القائمة.

- **حذف فهرس RemoveIndex:**
أرسل إلى هذه الخاصية كائنا من النوع PropertyDescriptor، لتقوم بحذف الخاصية التي يشير إليها، من مجموعة الفهارس المستخدمة في البحث في عناصر القائمة.

**إضافة عنصر جديد AddNew:**

لا تستقبل هذه الوسيلة أية معاملات، ولكنها تنشئ عنصرا جديدا من نفس نوع عناصر القائمة، وتضيفه إليها، وتعيد إليك كائنا Object يحمل مرجعا إلى هذا العنصر المضاف.

**تنفيذ الترتيب ApplySort:**

تقوم بترتيب عناصر القائمة، وهي تستقبل معاملين:

- كائن من النوع PropertyDescriptor، يوضح الخاصية التي سيتم ترتيب العناصر على أساسها.
- إحدى قيمتي المرقم ListSortDirection توضح اتجاه الترتيب.

**حذف الترتيب RemoveSort:**

تعيد عناصر القائمة إلى ترتيبها الأصلي الذي كانت عليه قبل استدعاء الوسيلة ApplySort.

**بحث Find:**

تقوم بالبحث في عناصر القائمة، وهي تستقبل معاملين:

- كائن من النوع PropertyDescriptor، يوضح الخاصية التي سيتم البحث عن قيمتها.
- كائن Object يحمل القيمة المراد البحث عنها.

كما تمتلك هذه الواجهة الحدث التالي:

**القائمة تغيرت ListChanged:**

ينطلق عند حدوث تغير في عناصر القائمة.. والمعامل الثاني e لهذا الحدث من النوع ListChangedEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| تعيد إحدى قيم المرقم ListChangedType التي توضح نوع التغيير الذي حدث، وهذه القيم هي:<br>- Reset: حدث تغير في عدد كبير من العناصر.<br>- ItemAdded: إضافة عنصر.<br>- ItemDeleted: حذف عنصر.<br>- ItemMoved: تغير موضع عنصر.<br>- ItemChanged: تغيرت قيمة عنصر.<br>- PropertyDescriptorAdded: إضافة واصف خاصية.<br>- PropertyDescriptorDeleted: حذف واصف خاصية.<br>- PropertyDescriptorChanged: تغيير واصف خاصية. | ListChangedType | |
| تعيد رقم العنصر الذي تغير في القائمة. | NewIndex | |
| تعيد رقم العنصر قبل تغيير موضعه في القائمة. | OldIndex | |
| تعيد كائنا من النوع PropertyDescriptor، يحتوي على واصف الخاصية، وذلك إذا كان التغيير قد حدث لواصف إحدى الخصائص. | PropertyDescriptor | |

## واجهة القائمة محددة النوع
## ITypedList Interface

تحصل هذه الواجهة على خصائص العنصر الذي سيتم الارتباط Binding بـــه، وهي تمتلك الوسيلتين التاليتين:

- **معرفة اسم القائمة GetListName:**
تعيد اسم القائمة التي سيتم الارتباط بها.

- **معرفة خصائص العنصر GetItemProperties:**
أرسل إلى هذه الخاصية مصفوفة من النـــوع PropertyDescriptor بهـــا الكائنات التي سيتم الارتباط بها في القائمة، لتعيد إليك مجموعة من النـــوع PropertyDescriptorCollection، بها خصائص هذه الكائنات.. لاحـــظ أنك لو أرسلت Nothing كمعامل، فستحصل على مجموعة بهـــا عنـــصر واحد فقط، وهو واصف الخاصية للقائمة نفسها.

## فئـــة مديـــر العـــرض
## DataViewManager Class

هذه الفئة تمثل الواجهتين IBindingList و ITypedList، كما أنها ترث الفئة MarshalByValueComponent، مما يتيح لك إضافتها إلى صينية مكونـــات النموذج، وإن كان عليك أن تضيفها أولا إلى صندوق الأدوات.
وتتيح لك هذه الفئة لك التحكم في كيفية عرض سجلات كل جداول في مجموعـــة البيانات.
ويمكن الحصول على مدير العرض من مجموعة البيانات، باســتخدام الخاصـــيّة DefaultViewManager، كالتالي:

**Dim DVM = Ds.DefaultViewManager**

ولحدث إنشاء هذه الفئة صيغتان:

1- الأولى بدون معاملات.
2- والثانية تستقبل مجموعة البيانات التي يتعامل معها مدير العرض.. مثال:

**Dim DVM As New DataViewManager (Ds)**

ويمتلك مدير العرض الخاصيتين التاليتين:

### مجموعة البيانات DataSet:
تقرأ أو تغيّر مجموعة البيانات التي يتعامل معها مدير العرض.

### إعدادات عرض البيانات DataViewSettings:
تعيد مجموعة إعدادات العرض DataViewSettingCollection، وهـــي مجموعة تمثل الواجهة ICollection، وتحتوي على إعـــدادات العـــرض الخاصة بجداول مجموعة البيانات، وكل عنصر من عناصرها من نوع الفئة DataViewSetting التي سنتعرف عليها لاحقا.

لاحظ أن هذه المجموعة للقراءة فقط، لهذا لا يمكنك إضافة أية عناصر إليها.. لكن كل جدول يضاف إلى مجموعة البيانات، يضيف عنصر إعدادات العرض الخاص به إلى هذه المجموعة تلقائيا.

ويمكنك الحصول على كائن إعدادات الجدول من هذه المجموعة، إما باستخدام رقم الجدول أو اسمه أو كائن الجدول DataTable الذي يشير إليه.. مثال:

```
Dim Vs As DataViewSetting =
    Ds.DefaultViewManager.DataViewSettings(0)
```

ويمتلك مدير العرض الوسيلة التالية:

- **إنشاء عرض بيانات CreateDataView:**

أرسل إلى هذه الوسيلة كائن الجدول DataTable لتنشئ عرضا View لسجلاته، تبعا للإعدادات الخاصة بهذا الجدول في مجموعة الإعدادات DataViewSettings.. وتعيد هذه الوسيلة كائن العرض DataView الذي تم إنشاؤه.. مثال:

```
Dim TblAuthors As DataTable = Ds.Tables("Authors")
Dim DV As DataView = Ds.DefaultViewManager.
    CreateDataView(TblAuthors)
```

## فئة إعدادات العرض DataViewSetting Class

تحتوي هذه الفئة على إعدادات العرض الافتراضيّة التي يتمّ استخدامها مع الجداول المعروضة.
وليس لهذه الفئة حدث إنشاء، ولكن يمكنك الحصول على نسخة منها خاصة بأحد الجداول من خلال مجموعة إعدادات العرض DataViewSettings كالتالي:

```
Dim Vs As DataViewSetting =
    Ds.DefaultViewManager.DataViewSettings("Authors")
```

وتمتلك هذه الفئة الخصائص:

### :Table

تعيد كائن الجدول DataTable الذي ينتمي إليه كائن الإعدادات الحالي.

### مدير العرض DataViewManager:

تعيد كائن مدير العرض DataViewManager الذي يحتوي كائن الإعدادات الحالي.

### مرشح حالة الصفوف RowStateFilter:

تحدد حالة الصفوف التي تريد عرضها، وهي تأخذ إحدى قيم المرقم DataViewRowState التالية:

| | |
|---|---|
| None | أي حالة. |
| Unchanged | عرض الصفوف التي لم تتغير. |
| Added | عرض الصفوف المضافة. |
| Deleted | عرض الصفوف المحذوفة. |
| OriginalRows | النسخة الأصلية للصفوف Original Version. |
| CurrentRows | النسخة الحالية للصفوف Current Version. |
| ModifiedOriginal | النسخة الأصلية للسجلات التي تم تعديلها. |
| ModifiedCurrent | النسخة الحالية للسجلات التي تم تعديلها. |

### مرشح الصفوف RowFilter:

تقرأ أو تغير الشرط الذي يتم على أساسه اختيار السجلات التي يعرضها كائن العرض.. ويتم تكوين الشرط في هذه الخاصية، بنفس الطريقة المستخدمة في تكوين شرط الخاصية DataRow.Expression.

والمرشح التالي يعرض الكتب التي يقل ثمنها عن 5 جنيهات:

**Vs.RowFilter = "Price < 5"**

### الترتيب Sort:

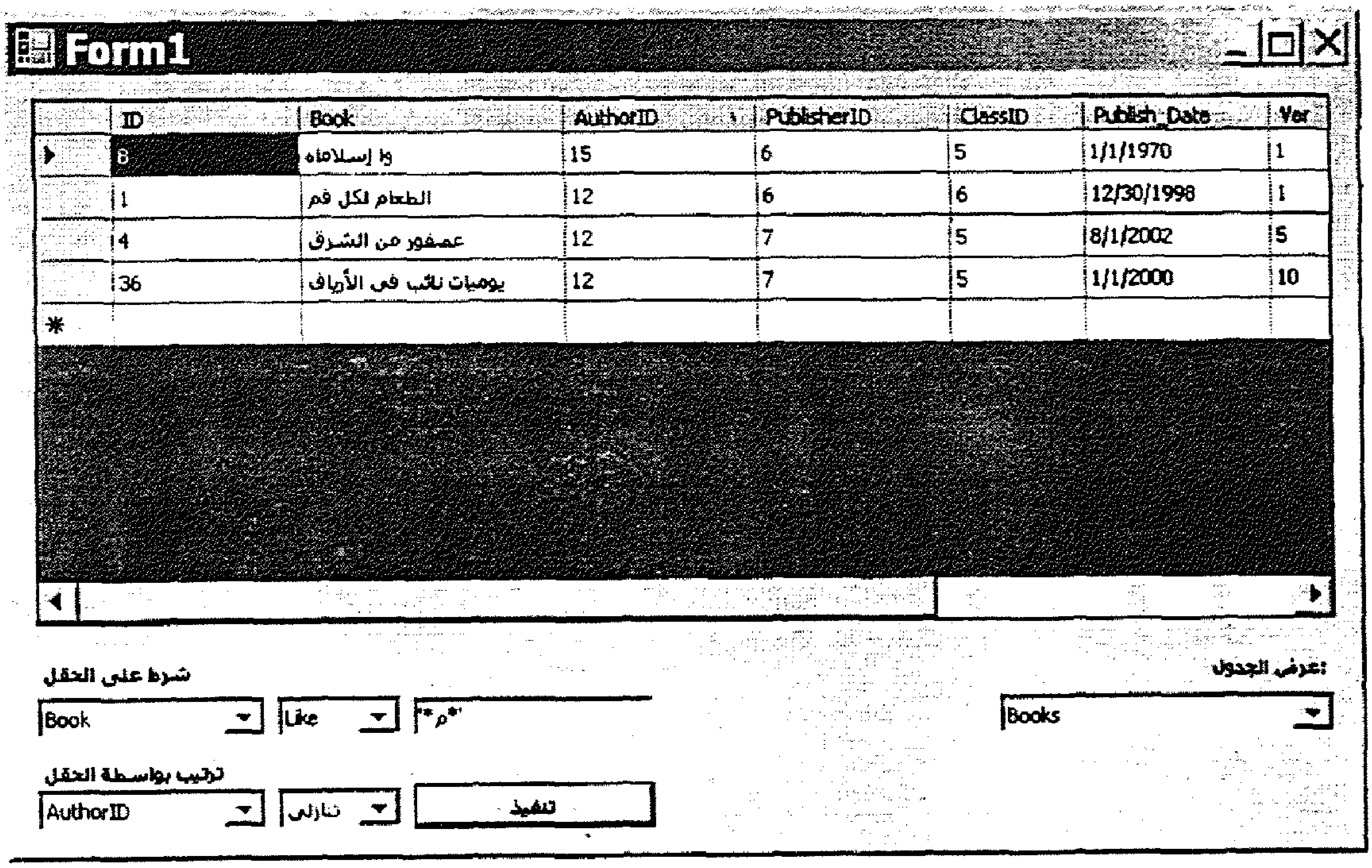
تحدد النص المستخدم في ترتيب السجلات المعروضة، وهو يتكون من اسم الحقل المستخدم في الترتيب، متبوعا باتجاه الترتيب.. مثلا: لترتيب صفوف جدول الكتب تنازليا على حسيب اسم الكتاب، استخدم القيمة التالية لهذه الخاصية:

**Vs.Sort = "Book DESC"**

### تطبيق الترتيب الافتراضي ApplyDefaultSort:

إذا جعلت قيمة هذه الخاصية True، فسيتم تطبيق الترتيب الافتراضي للصفوف (كما في الجدول الأصلي).. والقيمة الافتراضية لهذه الخاصية هي False، وفي هذه الحالة يتم تطبيق الترتيب الموضح في الخاصية Sort.

ويعتبر المشروع Views تدريبا جيدا على استخدام مدير العروض وإعدادات العرض.. في هذا المشروع نتيح للمستخدم عرض جداول قاعدة الكتب، باختيار الجدول الذي يريده من القائمة المنسدلة "عرض الجدول"، كما نتيح له اختيار مرشح الصفوف وطريقة الترتيب كما هو موضح في الصورة:



دعنا نفهم كيف يعمل هذا المشروع:

- في حدث تحميل النموذج Load، نستخدم موصلات الجداول لملء مجموعة البيانات، ثم نضيف الجداول إلى القائمة المركبة LstTables.
- في حدث تغير العنصر المحدد في قائمة الجداول SelectedIndexChanged، نعرض سجلات الجدول الذي اختاره المستخدم في جدول العرض، ونفرغ قائمة الحقول LstFields وقائمة الترتيب LstSort من محتوياتهما، ونضيف إلى كل منهما أسماء أعمدة الجدول الحالي.. ونظرا لأن المستخدم قد يريد عرض كل السجلات بدون ترتيب، فسنضيف إلى كل مجموعة عنصرا إضافيا اسمه (None) وسنجعله أول عنصر في كل منهما.

وللتحكم في طريقة عرض الجدول، سنستخدم إعدادات العرض الخاصة به.. ولتسهيل الكود في باقي البرنامج، سنضع كائن إعدادات الجدول الحالي في متغير معرف على مستوى النموذج اسمه ViewSettings:

```
Dim T As DataTable = LstTables.SelectedItem
ViewSettings = Ds.DefaultViewManager.
                DataViewSettings(T)
```

- في حدث تغير العنصر المحدد SelectedIndex في قائمة الحقول LstFields، سنعطل قائمة معاملات المقارنة LstOp ومربع نص القيمة TxtValue، وذلك إذا اختار المستخدم العنصر (None) وهو العنصر رقم صفر في القائمة، أما إذا اختار المستخدم أحد أعمدة الجدول، فسنفعّل قائمة المعاملات وجدول القيمة حتى يمكن استخدامهما في تكوين مرشح السجلات.. كل هذا يمكن فعله بالسطرين التاليين فحسب:

```
LstOp.Enabled = (LstField.SelectedIndex <> 0)
TxtValue.Enabled = LstOp.Enabled
```

- في حدث تغير العنصر المحدد SelectedIndex في قائمة حقل الترتيب LstSort: إذا اختار المستخدم العنصر (None) وهو العنصر رقم صفر في القائمة، فسنعطل قائمة اتجاه الترتيب LstSortOrder ونجعل للخاصية ApplyDefaultSort الخاصة بإعدادات العرض القيمة True لعرض السجلات بترتيبها الأصلي.. أما إذا اختار المستخدم أحد أعمدة الجدول، فسنفعّل قائمة اتجاه الترتيب ونضع False في الخاصية ApplyDefaultSort لنعرض السجلات بالترتيب الذي يريده المستخدم.. كل هذا يمكن فعله بالسطرين التاليين فحسب:

```
LstSortOrder.Enabled = (LstSort.SelectedIndex <> 0)
ViewSettings.ApplyDefaultSort = Not LstSortOrder.Enabled
```

- أخيرا، في حدث ضغط الزر تنفيذ، سنقوم بتنفيذ خيارات العرض التي اختارها المستخدم، حيث سنكون مرشح السجلات كالتالي:

```
If LstField.SelectedIndex = 0 Then
  ViewSettings.RowFilter = ""
Else
  ViewSettings.RowFilter = LstField.Text & " " &
      LstOp.Text & " " & TxtValue.Text.Trim
End If
```

لاحظ أن المستخدم هو المسئول عن كتابة القيمة المناسبة بالشكل الصحيح في مربع النص.. فعليه مثلا أن يكتب رقما في مربع النص إذا كان الحقل الذي اختاره رقميا، وأن يكتب نصا ويضعه بين علامتي التنصيص ' ' إذا كان الحقل يتعامل مع نصوص أو تواريخ.. وإذا اختار المعامل IN فعليه أن يفصل بين القيم بالعلامة , ويضع كل ما كتبه بين قوسين ( ).. وهكذا.
لاحظ أيضا أن وضع شرط خاطئ في الخاصية RowFilter لا يسبب خطأ في البرنامج، لكنه يجعل مدير الإعدادات يتجاهل هذه الإعدادات ولا ينفذها.
بعد هذا سنكون قيمة الخاصية Sort بالكود التالي:

```
If LstSort.SelectedIndex = 0 Then
  ViewSettings.Sort = ""
Else
  ViewSettings.Sort = LstSort.Text & " " &
    If(LstSortOrder.SelectedIndex = 0, "ASC", "DESC")
End If
```

أخيرا، سننشئ كائن عرض ونعرضه في جدول العرض.. لفعل هذا سنحصل على مدير العروض من كائن إعدادات العرض، ونستخدم الوسيلة CreateDataView الخاصة به كالتالي:

```
Dim Dv = ViewSettings.DataViewManager.
          CreateDataView(ViewSettings.Table)
Dgv.DataSource = Dv
```

وهكذا، وبهذا البرنامج البسيط، الذي لم نكتب به الكثير من الكود، صار باستطاعة المستخدم عرض جميع بيانات قاعدة الكتب، وبأي طريقة عرض يحبها!

## واجهة ربط قائمة العرض
## IBindingListView Interface

هذه الواجهة ترث الواجهة IBindingList، وهي تقدم إمكانيات متقدمة في ترشيح السجلات Filtering وترتيبها Sorting.

وتمتلك هذه الواجهة الخصائص التالية:

**المرشح Filter:**
تحدد الشرط الذي يتم على أساسه اختيار السجلات لعرضها.. مثل:
"Price > 5"

**واصفات الترتيب SortDescriptions:**
تعيد مجموعة للقراءة فقط من النوع ListSortDescriptionCollection، وهي تمثل واجهة القائمة IList، وكل عنصر من عناصرها من نوع الفئة ListSortDescription، التي سنتعرف عليها لاحقا.

**تدعم الترتيب المتقدم SupportsAdvancedSorting:**
تعيد True إذا كان مصدر البيانات يستطع ترتيب السجلات تبعا لقيم أكثر من عمود.

**تدعم الترشيح SupportsFiltering:**
تعيد True إذا كان مصدر البيانات يسمح بانتقاء بعض السجلات تبعا لشرط معين.

وتمتلك هذه الواجهة الوسيلة التالية:

**إزالة المرشح RemoveFilter:**
تلغي عملية الترشيح، وتعيد عرض جميع السجلات بدون مراعاة شرط الترشيح.

## فئة واصف ترتيب القائمة

## ListSortDescription Class

تحتوي هذه الفئة على المعلومات اللازمة لترتيب عناصر قائمة البيانات.
ويستقبل حدث إنشاء هذه الفئة معاملين:

- واصف الخاصية PropertyDescriptor التي سيتم الترتيب على أساسها.
- إحدى قيمتي المرقم ListSortDirection التي توضح اتجاه الترتيب.

وتمتلك هذه الفئة الخاصيتين التاليتين:

### واصف الخاصية PropertyDescriptor:

تقرأ أو تغير واصف الخاصية PropertyDescriptor التي سيتم الترتيب على أساسها.

### اتجاه الترتيب SortDirection:

توضح اتجاه الترتيب، وهي تأخذ إحدى قيمتي المرقم ListSortDirection وقد تعرفنا عليهما سابقا.

## فئة عرض البيانات DataView Class

هـــذه الفئـــة تمثـــل الواجهـــات IBindingListView و IBindingList و ITypedList، كما أنها ترث الفئة MarshalByValueComponent، مما يتيح لك إضافتها إلى صينية مكونات النموذج، وإن كان عليك أن تضيفها أولا إلى صندوق الأدوات.

ويمكنك استخدام كائن عرض البيانات للحصول على طريقـة عـرض مختلفـة لسجلات الجدول، سواء بترتيبها أو باختيار جزء من السجلات تبعا لشرط معين..
كما يمكنك ربط هذا الكائن بأدوات عرض البيانات كما سنرى في الفصل التالي، وبهذا تتحكم في طريقة عرض البيانات في برنامجك.

ويمكنك الحصول على كائن العرض الافتراضي للجـدول باسـتخدام الخاصـية DefaultView كالتالي:

**Dim Dv As DataView = TblAuthors.DefaultView**

ومن ثم يمكنك تغيير خصائصه لتغيير طريقة عرضه.

ولحدث إنشاء هذه الفئة الصيغ التالية:

1- الصيغة الأولى بدون معاملات.

2- الصيغة الثانية تستقبل كائن الجدول DataTable الذي سيتعامل معه كائن العرض.

3- الصيغة الثالثة تزيد على الصيغة السابقة بثلاثة معاملات، هي بالترتيب:

- نص يحتوي على شرط ترشيح السجلات RowFilter.
- نص يحتوي على شرط ترتيب السجلات Sort.
- إحدى قـيم المـرقم DataViewRowState، توضـح حالـة السجلات التي يتم عرضها.

والمثال التالي ينشئ كائن عرض، يعرض كتب توفيق الحكيم مرتبـة تـصاعديا على حسب أسمائها:

```
Dim Dv As New DataView(Ds.Books,
    "Parent(FK_Books_Authors).Author = 'توفيق الحكيم'",
    "Book", DataViewRowState.CurrentRows)
```

ويمكنك تجربة هذا المثال بضغط الزر "كتب توفيـق الحكـيم" فـي المـشروع Views.

وبالإضافة إلى ما تمثلـه مـن خـصائص الواجهـات IBindingListView و IBindingList و ITypedList، تمتلك فئة العـرض بعـض الخـصائص

المماثلة في اسمها ووظيفتها لخصائص كائن إعدادات العرض DataViewSetting، مثل:

- **الجدول Table**
- **مدير العرض DataViewManager**
- **مرشح الصفوف RowFilter**
- **مرشح حالة الصفوف RowStateFilter**
- **الترتيب Sort**
- **تطبيق الترتيب الافتراضي ApplyDefaultSort**

والكود التالي يجعل كائن العرض يحتوي على أسماء الكتب التي تبدأ أسماؤها بالحروف من الألف إلى التاء فقط، ويرتبها تنازليًا عن طريق اسم الكتاب ورقم المؤلف:

```
Dim T As DataTable = Ds.Tables("Books")
Dim Dv As DataView = T.DefaultView
Dv.RowFilter = " Book < 'ث' "
Dv.Sort = "Book, AuthorID DESC"
```

كما تمتلك فئة العرض الخاصية التالية:

**العنصر Item:**

هذه هي الخاصية الافتراضية، وهي تعيد كائن عرض الصف DataRowView الموجود في كائن العرض الحالي في الموضع المرسل كمعامل.. مثال:

```
Dim R As DataRowView = Dv(0)
```

لاحظ أن كائن العرض يعمل كقائمة، لهذا تستطيع المرور عبر كل صفوف كائن العرض باستخدام حلقة التكرار For Each كالتالي:

```
For Each R As DataRowView In Dv
    MsgBox(R("Book"))
Next
```

وتمتلك فئة العرض الوسائل التالية:

### إضافة سجل جديد AddNew:

تضيف سجلا جديدا إلى كائن العرض، تحتوي خاناته على القيم الافتراضية في الأعمدة التي لها قيم افتراضية، وعلى العدم DbNull في الأعمدة التي تسمح بهذا.. لاحظ أنّ هذا السجلّ سيضاف أيضا إلى الجدول الأصليّ أيضا الموجود في مجموعة البيانات، وعند القيام بعملية التحديث سيتم حفظه في قاعدة البيانات.. هذا يجعلك مطمئنا إلى أن السجلات الجديدة التي يضيفها المستخدم إلى جدول عرض DataGridView مرتبط بكائن عرض، ستتم إضافتها تلقائيا إلى مجموعة البيانات، وستعامل معاملة السجلات العادية.

وتعيد هذه الوسيلة كائن عرض الصف DataRowView الذي يشير إلى الصف الجديد الذي تمت إضافته، مما يتيح لك تغيير قيم خاناته.

### حذف Delete:

تحذف السجلّ الذي ترسل إليها رقمه كمعامل.. هذا السجل سيتم حذفه في الحال من كائن العرض، وستكون أمامنا حالتان بالنسبة للجدول الأصلي:

1- إذا كان السجل مضافا إلى الجدول (RowState = Added)، فسيتم حذفه نهائيا من الجدول، ولن يمكنك استعادته باستدعاء الوسيلة DataTable.RejectChanges.
2- إذا كان السجل أحد سجلات الجدول الأساسية، فسيظل في الجدول الأصلي مع جعل حالته محذوفا (RowState = Deleted).. ولو استدعيت الوسيلة DataTable.RejectChanges فستعيد السجل إلى كائن العرض DataView، مع تحويل حالته في الجدول الأصلي إلى Unchanged.

لاحظ أن ترتيب السجلات في كائن العرض قد يختلف عن ترتيبها المجموعة Rows في الجدول الأصلي، سواء بسبب إعادة ترتيبها، أو بسبب أخذ جزء فقط من سجلات الجدول الأصلي تحقق شرطا معينا.. لهذا عليك أن تستخدم الوسيلة Find لتبحث في كائن العرض عن السجل الذي تريد حذفه لتحصل على رقمه.. والمثال التالي يفترض أن Dv هو كائن عرض يعرض بعض سجلات جدول الكتب، ويستخدمه لحذف أحد الكتب:

```
Dv.Sort = "Book"
Dim I = Dv.Find("الطعام لكل فم")
If I <> -1 Then Dv.Delete(I)
```

### بحث Find:

أرسل إلى هذه الوسيلة كائنا Object، لتبحث عن القيمة التي يحملها في العمود المستخدم لترتيب السجلات (كما تحدده الخاصية Sort).. وتعيد هذه الوسيلة رقم أول سجلّ يحتوي على القيمة المطلوبة، وإذا لم تعثر على أيّ سجلّ، تعيد -1.

وتوجد صيغة ثانية لهذه الوسيلة، تستقبل مصفوفة كائنات، وذلك إذا كان كائن العرض يستخدم أكثر من عمود في عملية الترتيب.

وقد استخدمنا هذه الوسيلة في المشروع CustomDataSet.. في هذا المشروع يستطيع المستخدم إدخال أسماء الطلاب في النافذة الرئيسية، وضغط الرابط الموجود في عمود الدرجات لعرض نافذة فيها درجات الطالب في المواد المختلفة.. لفعل هذا، اتبعنا الخطوات التالية:

- حصلنا على كائن العرض الافتراضي الخاص بجدول الطلبة، باستخدام الخاصية DsStudents.Students.DefaultView، ومررنا إليه رقم صف التلميذ الحالي (ممثلا بالمتغير Idx)، لنحصل على كائن عرض هذا الصف DataRowView.

```
Dim StdRel = DsStudents.Students.ChildRelations(
      0).RelationName
Dim RowView = DsStudents.Students.
      DefaultView(Idx)
```

- أرسلنا إلى الوسيلة CreateChildView الخاصة بكائن الصف، اسم العلاقة التي تربط جدول التلاميذ بجدول الدرجات، لنحصل على كائن عرض يحتوي على درجات التلميذ الحالي، لنستخدمه كمصدر بيانات لجدول العرض الذي يعرض الدرجات.. هذا يجعل جميع الصفوف التي تضاف إلى هذا الجدول، تضع تلقائيا رقم هذا التلميذ في العمود StudentID:

```
Dim Grades = RowView.CreateChildView(StdRel)
```

- لو كانت هذه أول مرة نعرض فيها درجات هذا التلميذ، فسيظهر الجدول فارغا.. هذا غير مقبول، لأن علينا أن نعرض للمستخدم أسماء المواد، ليكتب هو درجات الطالب المناظرة لها مباشرة.. ونظرا لأننا نسمح للمستخدم بتحرير جدول المواد الدراسية بضغط الزر "عرض المواد"، فمن الممكن أن تضاف مواد جديدة لم نقم بإضافتها إلى نافذة الدرجات من قبل.. لهذا علينا أن نستخدم الوسيلة Find للتأكد من أن رقم المادة غير موجود، ومن ثم نضيفها.. السبب في هذا أن تكرار نفس المادة مع نفس الطالب سيؤدي إلى حدوث

خطأ بسبب قيد التفرد المفروض على هذين الحقلين.. لكن استخدام الوسيلة Find يوجب علينا أولا أن نستخدم الحقل SubjectID كمفتاح للترتيب في الخاصية Sort:

```
Grades.Sort = "SubjectID"
```

بعد هذا سنمر على كل مادة في جدول المواد، ونبحث عن رقمها في جدول الدرجات، ولو لم تكن موجودة (نتيجة البحث = -1)، علينا استخدام الوسيلة AddNew لإضافة صف جديد إلى كائن العرض.. هذا الصف يكون فارغا، ما عدا الحقل StudentID الذي يأخذ تلقائيا رقم التلميذ الذي نتعامل معه.. لهذا علينا نضيف رقم المادة إلى هذا الصف الجديد.. لاحظ أن اسم المادة سيضاف تلقائيا لأنه عمود محسوب مبني على قيمة الحقل StudentID.. لكن هذا لن يحدث طالما ظل هذا الصف الجديد هو الصف الحالي.. لحسن الحظ أن الصف الحالي يتغير تلقائيا إذا أضفنا صفا جديدا بعده، لكن المشكلة تظل في آخر صف نضيفه.. لهذا علينا استخدام الوسيلة EndEdit لإنهاء تحرير الصف الحالي.. هذا سيجبر العمود Subject على حساب قيمته، وعرض اسم المادة تلقائيا:

```
Dim SubjRows = DsStudents.Subjects.Rows
For i = 0 To SubjRows.Count - 1
    Dim SbjID = SubjRows(i)("ID")
    If Grades.Find(SbjID) = -1 Then
        Dim Rv = Grades.AddNew()
         Rv("SubjectID") = SbjID
         Rv.EndEdit()
    End If
Next
```

ومن الأذكى أن تخفي رقم التلميذ ورقم المادة عن المستخدم، لأنه سيتعامل فقط مع المادة ودرجة الطالب فيها.. وستجد هذا الكود كاملا في الإجراء ShowGrades الذي يتم استدعاؤه من الحدث CellContentClick في النموذج FrmStudent، ومن الحدث SelectedIndex في النموذج FrmGrades.

### بحث عن الصفوف FindRows:

مماثلة للوسيلة السابقة في صيغتيها، ولكنها تعيد مصفوفة من النوع DataRowView، تحتوي على كلّ السجلات التي تحتوي على القيمة المطلوبة.

### التحويل إلى جدول ToTable:

تنشئ جدولا جديدا وتنسخ إليه السجلات الموجودة في كائن العرض الحالي، وتعيد إليك كائن جدول DataTable يشير إليه.. ولهذه الوسيلة الصيغ التالية:

1- الصيغة الأولى بدون معاملات.

2- الصيغة الثانية تستقبل نصا لتستخدمه كاسم للجدول.

3- الصيغة الثالثة تستقبل معاملين:

- قيمة منطقية إذا جعلتها True فسيتم استبعاد الصفوف المكررة التي تتشابه قيم أي من خاناتها مع أي صفوف أخرى.
- مصفوفة نصية، تحتوي على أسماء الأعمدة التي تريد نسخها إلى الجدول.. وستظهر الأعمدة في الجدول بنفس ترتيبها في المصفوفة.. لاحظ أن هذه هي الطريقة الوحيدة التي تستطيع بها الحصول على جدول يختلف في عدد أعمدته وترتيبها عن الجدول الأصلي.

4- الصيغة الرابعة تحتوي على معاملات الصيغتين السابقتين معا.

## واجهة الكائن القابل للتحرير
## IEditableObject Interface

تقدم الوسائل اللازمة للتعامل مع كائن يمكن وضعه في حالة التحرير (مثل كائن عرض صف البيانات DataRowView)، مع القدرة على إلغاء التغييرات أو قبولها.
وتمتلك هذه الواجهة الوسائل التالية:

- **بدء التحرير BeginEdit:**
تبدأ عملية التحرير.

- **إلغاء التحرير CancelEdit:**
تلغي عملية التحرير وتعيد إلى الكائن قيمه الأصلية.

- **إنهاء التحرير EndEdit:**
تنهي عملية التحرير وتحفظ في الكائن القيم التي تم إدخالها أثناء عملية التحرير.

## واجهة معلومات خطأ البيانات
## IDataErrorInfo Interface

تقدم معلومات عن الخطأ الذي حدث في السجل، وهي تمتلك الخاصيتين التاليتين:

- **الخطأ Error:**
تعيد نصا يشرح الخطأ الذي حدث في السجل.

- **خطأ العنصر Item:**
أرسل إلى هذه الخاصية اسم العمود، لتعيد إليك نصا يشرح الخطأ الذي حدث في هذا العمود في السجل الحالي.

## واجهة التنبيه بتغير خاصية
## INotifyPropertyChanged Interface

تطلق الحدث PropertyChanged عند تغير قيمة خاصية معينة.. وتمتلك هـذه الواجهة الحدث التالي:

**الخاصية تغيرت PropertyChanged:**
ينطلق عندما يحدث تغيير في قيمة الخاصية.. والمعامل الثاني e لهذا الحدث من النـوع PropertyChangedEventArgs، وهـو يمتلـك الخاصـية PropertyName التي تعيد اسم الخاصية التي تغيرت.

## فئة عرض صف البيانات DataRowView Class

هـذه الفئـة تتمـل الواجهـات IEditableObject و IDataErrorInfo و INotifyPropertyChanged، وهي تحتوي على أحد السجلات المعروضـة في كائن العرض DataView.. ولا يوجد لهذه الفئة حدث إنـشاء، ولا يمكنـك الحصول على نسخة منها إلا من خلال كائن العرض.

وتمتلك هذه الفئة الخصائص التالية:

**عرض البيانات DataView:**
تعيد كائن العرض DataView الذي ينتمي إليه الصف الحالي.

**السجل Row:**
تعيد كائن سجل البيانات DataRow الذي يعرضه كائن عـرض الـصف الحاليّ.

**في حالة التحرير IsEdit:**
تعيد True إذا كان السجلّ الحاليّ يتمّ تحريره.

### هل هو جديد IsNew:

تعيد True إذا كان السجلّ الحاليّ جديدا، وذلك إذا تم إنشاؤه بواسطة الوسيلة DataView.AddNew.. مثال:

```
Dim Rv = Dv.AddNew()
MsgBox(Rv.IsNew) ' True
```

### العنصر Item:

هذه هي الخاصيّة الافتراضيّة Default Property، وهي تقرأ أو تغير قيمة إحدى خانات الصف الحالي، التي يحددها اسم العمود أو رقمه المرسل كمعامل.. والمثال التالي يعرض قيم كل الخانات الموجودة في كل صفوف كائن العرض Dv:

```
For Each R As DataRowView In Dv
    For I = 0 To Dv.Table.Columns.Count - 1
        MsgBox(R(I))
    Next
Next
```

### نسخة السجل RowVersion:

تعيد إحدى قيم المرقم DataRowVersion، التي توضح نسخة السجل المعروضة حاليا.

وتمتلك فئة عرض الصف الوسيلتين التاليتين:

### حذف Delete:

تحذف السجل الحالي من كائن العرض، ولها نفس تأثير الوسيلة DataView.Delete على السجل الأصلي في جدول البيانات.

### إنشاء عرض تابع CreateChildView:

تستخدم هذه الوسيلة إذا كان عارض الصف الحالي ينتمي إلى جدول أساسيّ يرتبط بعلاقة بجدول ثانويّ.. وتستقبل هذه الوسيلة اسم العلاقة أو كائن العلاقة DataRelation الذي يمثلها، وتعيد كائن عرض DataView يحتوي على كلّ سجلات الجدول الثانويّ التابعة لهذا السجلّ.. والمثال التالي يعيد كائن عرض يحتوي على كلّ الكتب التي ألفها أول مؤلف في جدول

الكتب، بافتراض أن العلاقة بين جدولي الكتب والمؤلفين اسمها AuthorsBooks:

```
Dim Rv As DataRowView
Rv = Ds.Tables("Authors").DefaultView(0)
Dim Dv As DataView
Dv = Rv.CreateChildView("AuthorsBooks")
```

وقد استخدمنا هذه الوسيلة في المشروع DataSetSample، لعرض كتب المؤلف المحدد حاليا.. فعند تغير السجل المحدد في جدول عرض المؤلفين، ينطلق الحدث DataGridView.RowEnter، وفيه نفعل ما يلي:

- نستخدم الخاصية e.RowIndex لمعرفة رقم الصف المحدد حاليا.
- نظرا لأن رقم الصف في جدول العرض، هو نفسه رقم الصف في جدول المؤلفين بسبب الربط بينهما Binding، فسنرسله إلى الخاصية الافتراضية DefaultView.Item للحصول على كائن عرض هذا الصف:

```
Dim TblAuthors = Ds.Tables("Authors")
Dim RowView = TblAuthors.DefaultView(e.RowIndex)
```

لاحظ أن رقم كل صف في جدول العرض يظل ثابتا مهما غير المستخدم ترتيب الصفوف المعروضة!

- نستخدم الوسيلة CreateChildView للحصول على كائن عرض يحتوي على كتب المؤلف المحدد حاليا، ونعرضها في جدول عرض الكتب الموجود في النصف السفلي من النافذة:

```
Dim DvBooks As DataView =
    RowView.CreateChildView("AuthorsBooks")
DgBooks.DataSource = DvBooks
```

لاحظ أنك عندما تبدأ تحرير السجل الجديد الموجود في نهاية جدول عرض الكتب، فإن الخانة AuthorID ستأخذ تلقائيا رقم المؤلف الذي ينتمي إليه كائن العرض الحالي.. هذا يريحك من كتابة أي كود إضافي، كما يسمح لك بإخفاء العمود AuthorID من الجدول دون قلق، فهو سيأخذ القيمة الصحيحة آليا دون أن يشغل المستخدم باله بهذا.. ولإخفاء هذا العمود، أضف هذا السطر إلى نهاية الكود السابق:

```
DgBooks.Columns("AuthorID").Visible = False
```

# -15-

# ربط البيانات

# Data Binding

في تطبيقات قواعد البيانات، كثيرا ما تحتاج إلى عرض بيانات أحد السجلات في أدوات موضوعة على النموذج، مع وجود أزرار للتحرك إلى السجل التالي أو السجل السابق، ليتمكن المستخدم من التحكم في السجل المعروض حاليا.
ونظرا لأن فعل هذا يدويا قد يحتاج إلى كود طويل ومرهق، فقد قدمت لك دوت نت آلية جاهزة تسمى ربط البيانات Data Binding، تتيح لك ربط الكائنات بأدوات الويندوز بأقل قدر من الكود.
ويسمى الكائن الذي يتم ربطه بالأداة باسم **مصدر البيانات Data Source**.. وتشمل مصادر البيانات الأنواع التالية:

**1- الكائنات البسيطة التي تحتوي على بعض الحقول Fields:**

مثل كائن الحجم Size Object الذي يحتوي على الحقلين Width و Height.. والمشروع BindingToObject المرفق بأمثلة هذا الكتاب يريك مثالا على هذا الربط.

**2- الكائنات المركبة التي تحتوي على عدة كائنات:**

كالمصفوفات Arrays التي تحتوي على أرقام أو نصوص أو كائنات محددة النوع، والمجموعات Collections التي تمثل واجهة القائمة IList.. وتعرض الأدوات في هذه الحالة عنصرا واحدا فقط في نفس اللحظة، وتقدم تقنية الربط الوسائل اللازمة للتحرك إلى العنصر السابق أو التالي.. والمشروع BindingToArray يريك مثالا على هذا الربط.

**3- الكائنات المعقدة التي تحتوي على مجموعات داخلية:**

وهي الكائنات التي تمثل الواجهة IBindingList أو الواجهة ITypedList، مثل مجموعة البيانات DataSet وجدول البيانات DataTable وعرض البيانات DataView ومدير عرض البيانات DataViewManager.. ونظرا لأن مصدر البيانات يحتوي على أكثر

من مجموعة داخلية (مجموعة البيانات مثلا تحتوي على أكثر من جدول وأكثر من علاقة)، فيجب أن نحدد الخاصية التي سنأخذ البيانات منها (كاسم الجدول مثلا).. وتسمى هذه الخاصية باسم **عنصر البيانات Data Member.**

وتعرض الأدوات في هذه الحالة سجلا واحدا فقط في نفس اللحظة، وتقدم تقنية الربط الوسائل اللازمة للتحرك إلى السجل السابق أو التالي.. والمشروع BindingToDataSet يريك مثالا على هذا الربط.

## واجهة المكون القابل للارتباط IBindableComponent Interfac

تمتلك هذه الفئة العناصر الأساسية اللازمة لربط الأدوات بمصادر البيانات، وهي:

**ارتباطات البيانات DataBindings:**

تعيد نسخة من مجموعة ارتباطات الأداة ControlBindingsCollection، تحتوي على كائنات الربط Binding Objects التي تستخدمها الأداة الحالية.. وسنتعرف على الفئة Binding بالتفصيل لاحقا.

**محتوى الربط BindingContext:**

تقرأ أو تغير كائن محتوى الربط BindingContext الذي تستخدمه الأداة.. وسنتعرف على الفئة BindingContext بالتفصيل لاحقا.

**تصفير الارتباطات ResetBindings:**

تقوم هذه الوسيلة بإنعاش القيم التي تعرضها الأداة من خلال الارتباط.. هذه الوسيلة ليست على درجة ملموسة من الأهمية.

**محتوى الربط تغير BindingContextChanged:**

ينطلق هذا الحدث عند تغير قيمة الخاصية BindingContext.

الجدير بالذكر أن فئة الأداة الأم Control Class التي ترثها جميع الأدوات تمثل الواجهة IBindableComponent، ومن ثم فهي تمتلك جميع العناصر السابقة.. هذا معناه أن جميع أدوات الويندوز تصلح للارتباط بمصادر البيانات.

وتسمى الأدوات التي يتم الارتباط بها باسم **الأدوات المرتبطة بالبيانات Data-Bound Controls**، ونظرا لأن كل أداة تمتلك العديد من الخصائص، فيجب عليك أن تحدد الخاصية التي تريدها أن تعرض البيانات.. ولا مانع من أن تربط أكثر من خاصية من خصائص الأداة، بأكثر من عنصر من عناصر مصدر البيانات.. مثلا: يمكنك ربط الخاصية Text الخاصة بزر الاختيار CheckBox بعنصر بيانات نصي وربط الخاصية Checked بعنصر بيانات منطقي Boolean.. وتسمى خاصية الأداة التي يتم الارتباط بها باسم **عنصر العرض Display Member**، لأنها تعرض قيمة خاصية الكائن.

| **ملخص:** |
|---|
| • **مصدر البيانات Data Source:**<br>هو الكائن الذي يحتوي على البيانات التي يتم ربطها بالأداة.. ومثال ذلك: الجدول Books في مجموعة البيانات.<br><br>• **عنصر البيانات Data Member:**<br>هو الخاصية التي يتم ربط قيمتها بالأداة.. ومثال ذلك العمود Book في جدول الكتب.<br><br>• **الأداة المرتبطة بالبيانات Data-Bound Control:**<br>هي الأداة التي ترتبط بكائن وتعرض بعض بياناته.. مثل مربع النص TextBox أو اللافتة.. لاحظ أن كل الأدوات تصلح لعرض البيانات، وعليك اختيار ما يناسب وظيفة برنامجك منها.<br><br>• **عنصر العرض Display Member:**<br>هو خاصية الأداة التي ترتبط بعنصر البيانات وتعرض قيمته، كالخاصية Text في مربع النص أو اللافتة.. لاحظ أن كثيرا من خصائص الأدوات تصلح كعناصر عرض، حتى لو لم يرَ المستخدم قيمتها، فأحيانا تريد حفظ قيمة من مصدر البيانات في الأداة لاستخدامها في وظيفة البرنامج، كأن تحفظ رقم المؤلف في الخاصية Tag بينما تعرض اسمه في الخاصية Text. |

وتستخدم في تقنية الربط مجموعة من الفئات الموجودة في نطاق الاسم System.Windows.Forms.. دعنا نتعرف على هذه الفئات.

## فئة مجموعة الارتباطات

## BindingsCollection Class

ترث هذه المجموعة فئة المجموعة الأساسية BaseCollection، وهي مجموعة تقليدية تمثل الواجهة ICollection، وتشتق منها العديد من المجموعات الخاصة بالارتباطات وأدوات ربط البيانات، والتي سنتعرف عليها لاحقا.
وتحتوي مجموعة الارتباطات BindingsCollection على كل الارتباطات التي تم إنشاؤها بين أداة معينة ومصادر البيانات المختلفة.. وكل عنصر من عناصر هذه المجموعة من نوع الفئة Binding التي سنتعرف عليها بعد قليل.
ويمكن الحصول على هذه المجموعة من خلال الخاصية BindingManagerBase.Bindings، كما سنرى بعد قليل.

ولا جديد في هذه المجموعة إلا امتلاكها للحدثين التاليين:

- **يتم تغيير المجموعة CollectionChanging:**
ينطلق هذا الحدث عند تغيير المجموعة، والمعامل الثاني e لهذا الحدث من النوع CollectionChangeEventArgs، وقد تعرفنا عليه من قبل عند التعرف على كائن مجموعة الجداول DataTableCollection.

- **تم تغيير المجموعة CollectionChanged:**
ينطلق هذا الحدث بعد حدوث التغيير في المجموعة فعلا.. والمعامل الثاني لهذا الحدث مماثل لمعامل الحدث السابق.

## فئة مجموعة ارتباطات الأداة

## ControlBindingsCollection Class

ترث هذه المجموعة الفئة BindingsCollection، ويمكن الحصول عليها من خلال الخاصية Control.Bindings.

وتمتلك هذه المجموعة الوسائل التقليدية للمجموعات، لكن الوسيلة Add الخاصة بها لها الصيغ التالية:

1- الصيغة الأولى تستقبل كائن الارتباط Binding المراد إضافته إلى المجموعة.

2- الصيغة الثانية تستقبل ثلاثة معاملات:

- اسم عنصر العرض، مثل "Text".
- الكائن Object الذي يعمل كمصدر للبيانات، مثل كائن الحجم Size.
- اسم عنصر البيانات.. فمثلا، لو أردت ربط خاصية الارتفاع الخاصة بكائن الحجم Size، فأرسل إلى هذا المعامل النص "Height".. ويمكنك إرسال نص فارغ "" إلى هذا المعامل، وفي هذه الحالة سترتبط الأداة بالنص الناتج من الوسيلة ToString الخاصة بالكائن.. وإذا كان الكائن يحتوي على كائنات متداخلة (مثل مجموعة البيانات التي تحتوي على جداول، وكل منها تحتوي على أعمدة)، فيجب عليك كتابة مسار الخاصية كاملا بدون اسم الكائن.. فمثلا: يمكنك الارتباط بحقل اسم الكتاب في جدول الكتب باستخدام المسار "Books.Book"، مع إرسال مجموعة البيانات نفسها إلى المعامل الثاني.

3- الصيغة الثالثة تزيد على الصيغة السابقة بمعامل رابع، إذا جعلت قيمته True فسيتم تطبيق التنسيق Format الخاص بك عند عرض العنصر في الأداة.

4- الصيغة الرابعة تزيد على الصيغة السابقة بمعامل خامس، يستقبل إحدى قيم المرقم DataSourceUpdateMode، لوضعها في الخاصية DataSourceUpdateMode وسنتعرف عليها بعد قليل.

5- الصيغة الخامسة تزيد على الصيغة السابقة بمعامل سادس، وهو يستقبل القيمة التي تريد وضعها في عنصر العرض إذا كان عنصر البيانات فارغا Nothing.. لاحظ أن هذا المعامل من النوع Object ليتيح لك إرسال أية قيمة تناسبك.

6- الصيغة السادسة تزيد على الصيغة السابقة بمعامل سابع، يتيح لك إرسال نص يحمل التنسيق Format الذي تريد تطبيقه على قيمة عنصر البيانات عند وضعها في عنصر العرض.

7- الصيغة السابعة تزيد على الصيغة السابقة بمعامل ثامن من نوع الواجهة IFormatProvider، لوضع قيمته في الخاصية FormatInfo التي سنتعرف عليها بعد قليل.

وتعيد إليك كل هذه الصيغ ـ ما عدا الصيغة الأولى ـ مرجعا إلى كائن الارتباط Binding الذي تم إنشاؤه وإضافته إلى المجموعة.

وتستطيع استخدام الوسيلة Add أيضا لإضافة ارتباط بكائن بسيط حتى لو كان مجرد متغير نصي، مثل:

```
Dim Name As String = "Mohammad"
TextBox1.DataBindings.Add("Text", Name, "")
```

هذا الكود سيجعل مربع النص TextBox1 يعرض النص Mohammad.

كما يمكنك استخدامها لإضافة ارتباط بكائن مكون من أكثر من عنصر، مثل الفئات والسجلات.. مثال:

```
Dim Sz As New Size(100, 200)
TextBox2.DataBindings.Add("Text", Sz, "Width")
```

هذا الكود سيجعل مربع النص TextBox2 يعرض الرقم 100.

ويمكنك استخدام الوسيلة Add لربط أكثر من مصدر بيانات بنفس الأداة.. هذا الكود صحيح:

```
TextBox1.DataBindings.Add("Text", Name, "")
TextBox1.DataBindings.Add("Tag", Sz, "Width")
```

لكن الوسيلة Add ستسبب خطأ في البرنامج لو حاولت ربط أكثر من مصدر بيانات بنفس عنصر العرض في نفس الأداة.. هذا الكود خاطئ:

```
TextBox1.DataBindings.Add("Text", Name, "")
TextBox1.DataBindings.Add("Text", Sz, "Width")
```

لهذا قبل أن تغير ارتباط الخاصية، عليك أن تزيل كائن الارتباط أولا من المجموعة باستخدام الوسيلة Remove كالتالي:

```
Dim Bnd = TextBox1.DataBindings.Add("Text", Name, "")
TxtName.DataBindings.Remove(Bnd)
TextBox1.DataBindings.Add("Text", Sz, "Width")
```

لاحظ كذلك أن عنصر البيانات يجب أن يكون خاصية وليس متغيرا.. بمعنى آخر: لا يمكنك ربط الأداة بحقل Field من حقول الكائن.. تذكر أن الحقل هو متغير عام Public Variable معرف على مستوى الفئة، مثل:

```
Class Student
    Public ID, Age As Integer
    Public Name As String
End Class
```

الآن لو عرفت كائنا من فئة الطالب وليكن:

```
Dim Std As New Student With { .ID = 1, .Age = 15,
    .Name = "Ahmad"}
```

فإن محاولة ربط أي حقل خاص بالكائن Std (وليكن الحقل Name) بأية أداة ستؤدي إلى حدوث خطأ في البرنامج:

```
TxtId.DataBindings.Add("Text", Std, "Name")
```

لهذا عليك تغيير الحقول العامة في فئة الطالب لتصير خصائص.. كل ما عليك هو استخدام الخصائص ذاتية التعريف Auto Implemented Properties التي قدمتها فيجيوال بيزيك دوت نت 1010 بإضافة الكلمة Property قبل اسم المتغير، دون الحاجة إلى كتابة كود الخصائص كالتالي:

```
Class Student
    Public Property ID As Integer
    Public Property Age As Integer
    Public Property Name As String
End Class
```

الآن لو جربت ربط أي خاصية من خصائص الكائن Std، فسيعمل كل شيء على ما يرام.. والمشروع BindingToObject يريك الكود الكامل لربط خصائص فئة الطالب بمربعات النصوص.. وعليك عند فحص هذا المشروع أن تلاحظ ما يلي:

1- أن أي تغيير تجريه على مربعات النصوص سيؤثر على الكائن الأصلي Std.. ولو جربت تغيير بيانات الطالب في مربعات النصوص وضغط الزر "بيانات الطالب" فستعرض الرسالة القيم الموجودة في مربعات النصوص، رغم أن الكود المكتوب في حدث ضغط الزر يعرض بيانات المتغير Std.

2- إذا حاولت أن تكتب قيمة خاطئة في أي مربع نص ككتابة حروف في مربع النص الخاص برقم الطالب، فسيتم إلغاؤها بمجرد مغادرته.. هذا معناه أن تقنية الربط تجيز البيانات تلقائيا قبل مغادرة الأداة، فإذا كانت ستسبب خطأ عند وضعها في عنصر البيانات يتم إلغاء القيمة من عنصر العرض وإعادته إلى قيمته السابقة.

كما يمكنك أيضا الارتباط بكائنات معقدة، مثل مصفوفة تحتوي على عناصر من نوع فئة التلميذ، أو كائنات أكثر تعقيدا مثل مجموعة البيانات التي تحتوي على

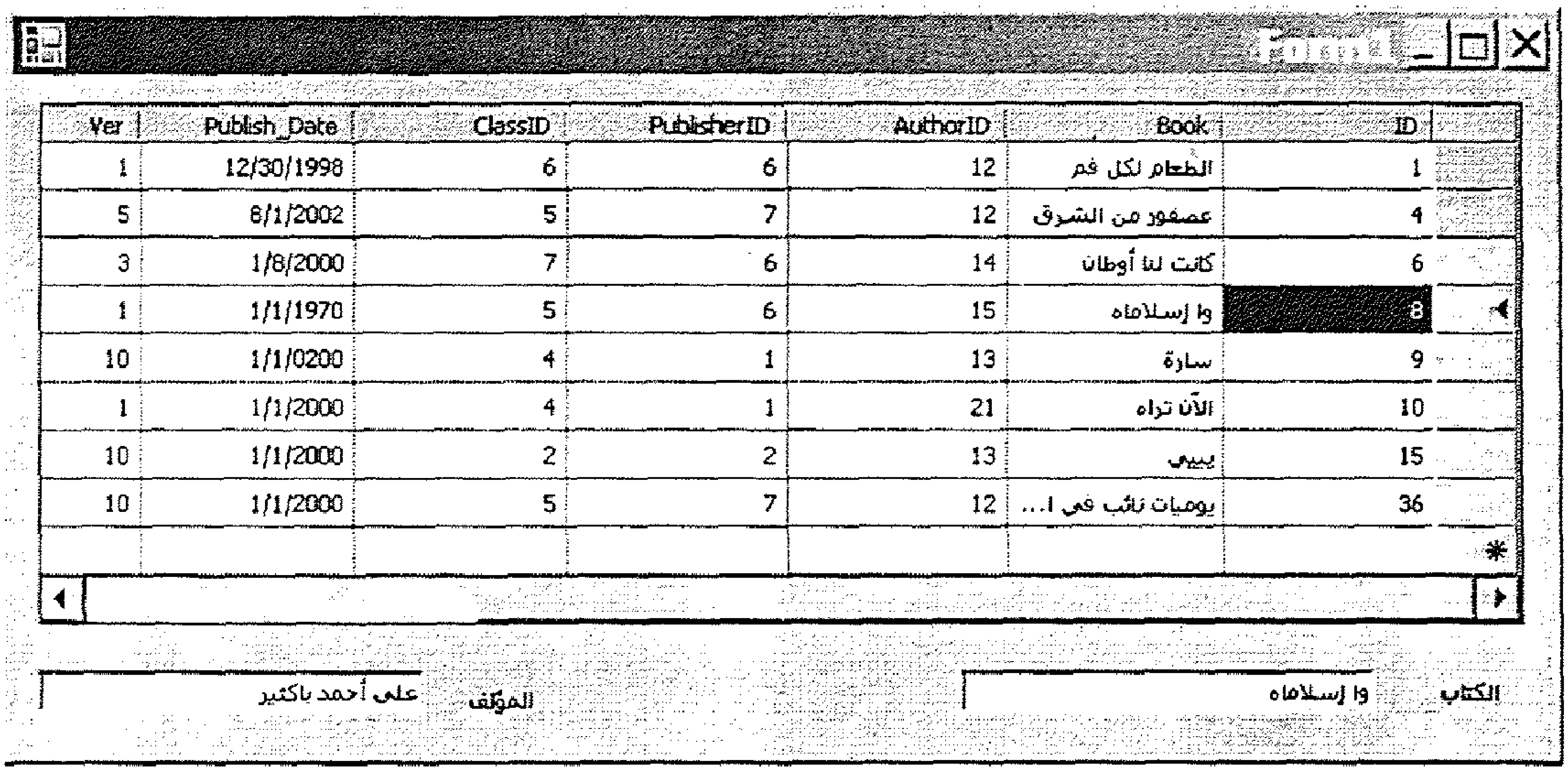
جداول، بكل منها أعمدة يعتبر كل عمود منها مصفوفة (لأن به صفوفا) ويصلح كعنصر بيانات.. والمثال التالي ينشئ كائن ارتباط ويضيفه إلى مجموعة ارتباطات مربع النص، ليجعله يعرض اسم الكتاب الحالي في جدول الكتب:

```
Dim B As Binding = TextBox1.DataBindings.Add("Text",
    Ds, "Books.Book")
```

ويمكن فعل نفس الشيء أيضا بالكود التالي:

```
Dim B As Binding = TextBox1.DataBindings.Add("Text",
    Ds.Tables("Books"), "Book")
```

ويريك المشروع BindingTextBox مثالا طريفا على هذا، حيث سنجعل مربع نص يعرض اسم الكتاب الحالي، ومربع نص آخر يعرض اسم مؤلفه.. وسنعرض جدول الكتب كله في جدول عرض DataGridView الذي سنتعرف على طريقة ربطه لاحقا.



| Ver | Publish_Date | ClassID | PublisherID | AuthorID | Book | ID |
|---|---|---|---|---|---|---|
| 1 | 12/30/1998 | 6 | 6 | 12 | الطعام لكل فم | 1 |
| 5 | 6/1/2002 | 5 | 7 | 12 | عصفور من الشرق | 4 |
| 3 | 1/8/2000 | 7 | 6 | 14 | كانت لنا أوطان | 6 |
| 1 | 1/1/1970 | 5 | 6 | 15 | وا إسلاماه | 8 |
| 10 | 1/1/0200 | 4 | 1 | 13 | سارة | 9 |
| 1 | 1/1/2000 | 4 | 1 | 21 | الآن تراه | 10 |
| 10 | 1/1/2000 | 2 | 2 | 13 | بيبي | 15 |
| 10 | 1/1/2000 | 5 | 7 | 12 | يوميات نائب في ا... | 36 |



الجميل في الأمر أن المستخدم كلما انتقل من صف إلى آخر في جدول العرض، يعرض مربعا النص اسم الكتاب الموجود في هذا الصف واسم مؤلفه تلقائيا، وبدون أن نكتب أي كود!.. السبب في هذا أن جدول العرض يغير الصف الحالي في كائن الارتباط، فيقوم تلقائيا بتحديث القيم المعروضة في جميع الأدوات المرتبطة به!.. لكن لكي تعمل هذه الطريقة، يجب أن يكون مصدر البيانات المرتبط به جدول العرض هو نفسه مصدر بيانات مربعي النص.. هكذا مثلا:

```
DataGridView1.DataSource = Ds
DataGridView1.DataMember = "Books"
TxtBook.DataBindings.Add("Text", Ds, "Books.Book")
TxtAuthor.DataBindings.Add("Text", Ds, "Books.Author")
```

أو يمكن استخدام جدول الكتب كمصدر بيانات للاختصار:

```
DataGridView1.DataSource = Ds.Books
TxtBook.DataBindings.Add("Text", Ds.Books, "Book")
TxtAuthor.DataBindings.Add("Text", Ds.Books, "Author")
```

لكن الكود التالي لن يجعل البرنامج يعمل بشكل صحيح، لأن مصدر بيانات جدول العرض (وهو Ds.Books) مختلف عن مصدر بيانات مربعي النص (وهو Ds):

```
DataGridView1.DataSource = Ds.Books
TxtBook.DataBindings.Add("Text", Ds, "Books.Book")
TxtAuthor.DataBindings.Add("Text", Ds, "Books.Author")
```

كذلك فإن الكود التالي أيضا لن يجعل البرنامج يعمل بشكل صحيح، لأن مصدر بيانات جدول العرض (وهو Ds) مختلف عن مصدر بيانات مربعي النص (وهو Ds.Books):

```
DataGridView1.DataSource = Ds
DataGridView1.DataMember = "Books"
TxtBook.DataBindings.Add("Text", Ds.Books, "Book")
TxtAuthor.DataBindings.Add("Text", Ds.Books, "Author")
```

وتمتلك مجموعة الارتباطات الخصائص الجديدة التالية:

**الأداة Control:**

تعيد الأداة التي تنتمي إليها مجموعة الارتباطات الحاليّة.

**المكون القابل للارتباط BindableComponent:**

تعيد واجهة المكون القابل للارتباط IBindableComponent التي تنتمي إليها مجموعة الارتباطات الحالية.

**الطريقة الافتراضية لتحديث مصدر البيانات DefaultDataSourceUpdateMode:**

تحدد القيمة الافتراضية للخاصية DataSourceUpdateMode لكل كائن ربط في المجموعة، وهي تأخذ إحدى قيم المرقم DataSourceUpdateMode التي سنتعرف عليها لاحقا.

# فئة الارتباط Binding Class

تقوم هذه الفئة بربط كائن يعمل كمصدر بيانات، بإحدى الأدوات، بحيث يأخذ عنصر العرض في الأداة قيمة عنصر البيانات في الكائن تلقائيا، وتغير قيمة أحدهما كلما تغيرت قيمة الآخر.

ولحدث إنشاء هذه الفئة نفس صيغ الوسيلة ControlBindingsCollection.Add، ما عدا الصيغة الأولى التي تستقبل كائن ارتباط Binding.

افترض أن لدينا مصفوفة أعداد صحيحة معرفة على مستوى النموذج كالتالي:

```
Dim A() As Integer = {1, 2, 3, 4}
```

سنعرّف الآن كائنا يربط الخاصيّة Text في مربّع النصّ بعناصر هذه المصفوفة:

```
Dim B As New Binding("Text", A, "")
TextBox1.DataBindings.Add(B)
```

لاحظ أن حدث الإنشاء استقبل ثلاثة معاملات:

- اسم خاصية الأداة وهي هنا "Text".
- مجموعة البيانات وهي هنا المصفوفة A.
- اسم عنصر العرض وقد تركناه فارغا لربط الكائن نفسه (المصفوفة).. لكن لو كنا نتعامل مع سجل الطالب Structure Student مثلا (كما فعلنا في المشروع BindingToArray)، فيمكن أن نرسل إلى هذا المعامل اسم أي حقل من حقوله، كالاسم "Name" أو العمر "Age".. أو غير ذلك.

وتمنحك الفئة Binding الخصائص التالية:

### المكون القابل للارتباط BindableComponent:

تعيد واجهة المكون القابل للارتباط IBindableComponent التي تمثلها الأداة التي ينتمي إليها الارتباط الحالي.

### الأداة Control:

تعيد الأداة Control التي ينتمي إليها الارتباط الحاليّ.

### اسم الخاصية PropertyName:

تقرأ أو تغير اسم عنصر العرض.

### مصدر البيانات DataSource:

تعيد الكائن Object الذي يعمل كمصدر للبيانات في هذا الارتباط.

### معلومات عنصر الربط BindingMemberInfo:
تعيد نسخة من سجل معلومات عنصر الربط BindingMemberInfo Structure، تحتوي على معلومات حول عنصر البيانات.. وسنتعرف على هذا السجل لاحقا.

### أساس مدير الربط BindingManagerBase:
تعيد كائن أساس الارتباط BindingManagerBase، الذي يتيح لك التحكم في الارتباط الحاليّ.. وسنتعرف على الفئة BindingManagerBase بالتفصيل لاحقا.

### هل هو مرتبط IsBinding:
تعيد True إذا كان الارتباط فعالا، وتعيد False إذا تم إيقاف الارتباط عن العمل مؤقتا.

### طريقة تحديث الأداة ControlUpdateMode:
تحدد كيفية تحديث عنصر العرض، عندما تتغير قيمة عنصر البيانات.. وتأخذ هذه الخاصية إحدى قيمتي المرقم ControlUpdateMode التاليتين:

| | |
|---|---|
| Never | لا يتم تحديث قيمة عنصر العرض عند تغير قيمة عنصر البيانات. |
| OnPropertyChanged | يتم تحديث قيمة عنصر العرض فور تغير قيمة عنصر البيانات.. هذه هي القيمة الافتراضية. |

### القيمة الفارغة NullValue:
تحدد القيمة التي ستوضع في عنصر العرض، إذا كانت لعنصر البيانات القيمة Nothing.. لاحظ أن هذه الخاصية ستكون بلا فائدة إذا كانت للخاصية ControlUpdateMode القيمة None.

## طريقة تحديث مصدر البيانات DataSourceUpdateMode:

تحدد كيفية تحديث قيمة عنصر البيانات عند تغير قيمة عنصر العرض، وهي تأخذ إحدى قيم المرقم DataSourceUpdateMode التالية:

| | |
|---|---|
| Never | لا يتم تحديث عنصر البيانات.. هذا يعني أن أي تغيير يحدث في عنصر العرض (كأن يكتب المستخدم في مربع النص مثلا) لن يؤثر على قيمة عنصر البيانات. |
| OnProperty Changed | يتم تغيير قيمة عنصر البيانات بمجرد تغيير قيمة عنصر العرض.. فمثلا لو كتب المستخدم في مربع النص، فإن ما كتبه يوضع فورا في عنصر البيانات. |
| OnValidation | لا يتم نقل التغيير من عنصر العرض إلى عنصر البيانات إلا عند انطلاق حدث تمام التحقق من الصحة Control.Validated، وذلك عند مغادرة الأداة أو عرض عنصر آخر من عناصر الكائن (إذا كان الكائن يحتوي على مجموعة من العناصر مثل المصفوفات).. هذه هي القيمة الافتراضية. |

## القيمة الفارغة لمصدر البيانات DataSourceNullValue:

تحدد القيمة التي ستوضع في عنصر البيانات، إذا كانت لعنصر العرض القيمة Nothing.. لاحظ أن هذه الخاصية ستكون بلا فائدة إذا كانت للخاصية DataSourceUpdateMode القيمة None.

## نص التنسيق FormatString:

تستقبل نصا يعرف صيغة التنسيق الذي سيستخدم لتنسيق البيانات قبل وضعها في عنصر العرض.. ويمكنك استخدام نفس صيغ تنسيق الأرقام والتواريخ التي تعرفنا عليها في ملاحق كتاب برمجة إطار العمل.

## تفعيل التنسيق FormattingEnabled:

إذا جعلت قيمة هذه الخاصية True فسيتم تطبيق التنسيق الموضح في الخاصية FormatString على البيانات قبل وضعها في عنصر العرض.

**معلومات التنسيق FormatInfo:**

تقرأ أو تغير واجهة مزود التنسيق IFormatProvider التي يستخدمها كائن الارتباط.. لقد عرفنا في كتاب برمجة إطار العمل أن الفئات NumberFormatInfo، DateTimeFormatInfo، CultureInfo تمثل واجهة مزود التنسيق IFormatProvider، لكن بالنسبة لهذه الخاصية، يمكنك استخدام نسخة من الفئة CultureInfo للتحكم في اللغة والثقافة التي ستستخدم عند تنسيق البيانات.. وفي الوضع الافتراضي تتعامل هذه الخاصية مع الثقافة (اللغة) المعرفة على جهاز المستخدم.

ويمتلك كائن الربط الوسيلتين التاليتين:

**قراءة القيمة ReadValue:**

تجبر الأداة على عرض قيمة عنصر البيانات.

**كتابة القيمة WriteValue:**

تجبر الأداة على وضع قيمة عنصر العرض، في عنصر البيانات.

ويمنحك كائن الارتباط الأحداث التالية:

**تنسيق Format:**

ينطلق قبل كتابة قيمة عنصر البيانات في عنصر العرض، ليسمح لك بتنسيق البيانات قبل أن تعرضها الأداة.

والمعامل e لهذا الحدث من النوع ConvertEventArgs، وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| | DesiredType | تعيد كائن النوع Type، الذي يمثل نوع بيانات عنصر العرض. |
| | Value | تقرأ أو تغيير البيانات التي سيتمّ وضعها في عنصر العرض.. هذه الخاصية من النوع Object للتعامل مع أنواع البيانات المختلفة. |

مثلا، يمكنك أن تعرض البيانات في الأداة بتنسيق التاريخ القصير كالتالي:

**e.Value = CType(e.Value, Date).ToString("d/MM/yy")**

### تحويل Parse:

ينطلق عندما تتغيّر قيمة عنصر العرض.. فإذا كنت قد غيرت تنسيق البيانات باستخدام الحدث Format، فاستخدم هذا الحدث لاستخلاص البيانات الأصليّة وإعادتها إلى النوع المناسب لوضعها في خاصية الكائن.

والمعامل e لهذا الحدث مماثل لذلك الخاصّ بالحدث السابق.. انظر كيف نستعيد التاريخ من النصّ الذي نسقناه في الحدث السابق:

```
e.Value = Date.Parse(e.Value.ToString)
```

### انتهى الربط BindingComplete:

ينطلق هذا الحدث بعد وضع قيمة عنصر البيانات في عنصر العرض، أو بعد وضع قيمة عنصر العرض في عنصر البيانات.. والمعامل الثاني e لهذا الحدث من النوع BindingCompleteEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | Binding | تعيد كائن الارتباط Binding الذي أطلق الحدث. |
| | Binding Complete Context | تخبرك باتجاه عملية الربط، وهي تعيد إحدى قيمتي المرقم BindingCompleteContext التاليتين:<br>- ControlUpdate: يتم تحديث الأداة.<br>- DataSourceUpdate: يتم تحديث مصدر البيانات. |
| | Binding Complete State | توضح حالة عملية الربط، وهي تعيد إحدى قيم المرقم BindingCompleteState التالية:<br>- Success: نجحت عملية الربط.<br>- DataError: فشلت عملية الربط بسبب خطأ في البيانات.. في هذه الحالة يرفض مصدر البيانات أو الأداة القيمة الجديدة، لكن لا يحدث خطأ في البرنامج.<br>- Exception: فشلت عملية الربط وحدث خطأ في البرنامج. |
| | ErrorText | تعيد نصا يصف الخطأ الذي حدث في عملية الربط. |
| | Exception | تعيد كائن الاستثناء Exception الذي يحتوي على تفاصيل الخطأ الذي حدث عند ربط البيانات. |

والمثال التالي يعرف كائن ربط Binding Object بين الخاصيّة Text لمربّع نصّ، وبين العمود Book في جدول الكتب في مجموعة البيانات Ds:

```
Dim B As New Binding("Text", Ds, "Books.Book")
```

هذا الارتباط لن يأخذ حيّزا من التنفيذ إلا إذا أضفناه إلى مجموعة الارتباطات DataBindings الخاصّة بمربّع النصّ، كالتالي:

```
TextBox1.DataBindings.Add (B)
```

وسنرى لاحقا كيف نتحرك بأنفسنا عبر صفوف جدول الكتب المختلفة، لنعرض أسماء الكتب المختلفة في مربع النص.. وعلى كل حال، ستجد هذا مطبقا في المشروع ViewAndEditBooks.

## سجل معلومات عنصر الربط

## BindingMemberInfo Structure

يحتوي هذا السجل على معلومات عن عنصر البيانات.. ولا تتضح فائدة هذا السجل عند الارتباط بكائنات بسيطة، وإنما تتضح عند الارتباط مع كائنات تحتوي على كائنات متداخلة.. في هذه الحالة عليك إرسال مسار الكامل لاسم الخاصية التي تعمل كعنصر بيانات، إلى حدث إنشاء هذا الكائن.. مثال:

**Dim Bmi As New BindingMemberInfo ("Books.Book")**

حيث Books هو اسم جدول الكتب، و Book هو الحقل الذي سيتم عرض قيمته في الأداة.. لاحظ أن هذا المسار لا يحتوي على اسم الكائن (وهو مجموعة البيانات Ds في مثالنا هذا)، فهذه المعلومة موجودة في كائن الربط Binding الذي ينتمي إليه السجل BindingMemberInfo.

ويمكنك أيضا الحصول على نسخة من هذا السجل باستخدام الخاصية Binding.BindingMemberInfo.

ويمتلك هذا السجل الخصائص التالية:

### عنصر الربط BindingMember:

تعيد المسار الكامل لعنصر البيانات... وفي المثال الذي ضربناه ستعيد هذه الخاصية النص: Books.Book .

### حقل الربط BindingField:

تعيد اسم عنصر البيانات (بدون المسار الكامل).. هذا يعني أنها ستعيد النص Book في المثال السابق.

### مسار الربط BindingPath:

تعيد مسار عنصر البيانات (بدون اسم الخاصية).. ولو استخدمنا هذه الخاصية مع المثال السابق، فستعيد النص Books.

ولو كان الكائن بسيطا وعنصر البيانات ليس له مسار، ففي هذه الحالة ستكون القيمة العائدة هي اسم الخاصية، مثل Width لو كان الارتباط بكائن الحجم Size.

## فئة محتوى الربط BindingContext Class

هذه الفئة تمثل واجهة المجموعة ICollection، وهي تعمل كمجموعة للقراءة فقط تحتوي على كائنات مدير الربط BindingManagerBase الخاصة بأداة معينة.. وسنتعرف على الفئة BindingManagerBase بعد قليل.
ويمكنك الحصول على نسخة من هذه الفئة باستخدام الخاصية Control.BindingContext.. والمثال التالي يعيد إليك مجموعة تحتوي على كائنات أساس مدير الربط للنموذج:

**Dim BC As BindingContext = Me.BindingContext**

لاحظ أن مجموعة محتوى الربط BindingContext الخاصة بالأداة الحاوية تحتوي على كل كائنات مدير الربط BindingManagerBase الخاصة بكل الأدوات الموجودة على هذه الأداة الحاوية.. هذا يعني أن محتوى الربط BC في المثال السابق سيحتوي على كائنات أساس الربط لكل الأدوات الموضوعة على النموذج (بشرط أن تكون داخلة في ارتباطات).. هذا يفيدك في تسهيل كتابة الكود عندما توجد العديد من الأدوات المرتبطة على النموذج.

وتمتلك هذه المجموعة الخاصية التالية:

### العنصر Item:

هذه هي الخاصيّة الافتراضيّة، وهي تعيد مدير الربط BindingManagerBase الموجود في المجموعة تبعا للمعاملات المرسلة.. ولهذه الخاصية صيغتان:

1. الصيغة الأولى لها معامل من النوع Object، يستقبل مصدر البيانات الذي تريد الحصول على مدير الربط الخاص به.. والكود التالي يوقف الارتباطات بين مجموعة البيانات Ds وكل الأدوات الموجودة على النموذج:

**Me.BindingContext(Ds).SuspendBinding( )**

2. الصيغة الثانية تزيد على الصيغة السابقة بمعامل نصي، يستقبل مسار عنصر البيانات.. هذا مفيد إذا كان الكائن يحتوي على العديد من مصادر البيانات وتريد التعامل مع واحد منها فقط.. وفي المشروع BindingToDataSet استخدمنا الجملة التالية للحصول على مدير الربط:

**Dim Bm = Me.BindingContext(DsBooks, "Books")**

لاحظ أن مدير الربط Bm في هذه الحالة يتعامل مع جدول الكتب، لهذا نستطيع استخدام الخاصية Bm.Position للتحكم في الكتاب المعروض حاليا في الأدوات.. بينما لو استخدمت الجملة التالية:

**Dim Bm = Me.BindingContext(DsBooks)**

فستحصل على مدير ربط يتعامل مع مجموعة البيانات نفسها، ونظرا لأنها تحتوي على العديد من القوائم الداخلية (مثل مجموعة الجداول ومجموعة العلاقات وغيرهما)، فلن يستطيع مدير الربط Bm في هذه الحالة التعامل مع جدول الكتب، وستشير الخاصية Bm.Count إلى أن هناك عنصرا واحدا فقط في مدير الربط (وهو مجموعة البيانات نفسها)، ولهذا لن تستطيع الانتقال إلى سجلات أخرى باستخدام الخاصية Bm.Position.

كما تمتلك هذه المجموعة الوسيلتين التاليتين:

### تحتوي على Contains:

تعيد True إذا كانت المجموعة تحتوي على مدير الربط المحدد في المعاملات.. ولهذه الوسيلة نفس صيغتي الخاصية الافتراضية Item.

### S تحديث الربط UpdateBinding:

أرسل إلى هذه الوسيلة مجموعة محتوى الربط BindingContext، وكائن الربط Binding الذي تريد إزالته من مجموعة أخرى وإضافته إلى المجموعة المحددة في المعامل الأول.

## فئة أساس مدير الربط BindingManagerBase Class

هذه الفئة أساسية مجردة Abstract Base Class، ومنها تشتق الفئات التي تعمل كمدير للربط.. والفئات التاليتان ترثان هذه الفئة:

1. فئة مدير التسلسل CurrencyManager Class.
2. فئة مدير الخاصية PropertyManager Class.

ولتعريف متغير من هذا الكائن، استخدم الصيغة التالية:

**Dim BM As BindingManagerBase**

ولوضع نسخة جديدة من مدير الربط في هذا المتغير، يمكنك استخدام الخاصية BindingManagerBase من كائن الارتباط Binding.. مثال:

**Dim Bnd = TextBox1.DataBindings.Add("Text", Obj, "")**
**BM = Bnd.BindingManagerBase**

لاحظ أن نوع المدير الذي سيوضع في المتغير BM يتوقف على نوع الكائن.. فلو كان كائنا بسيطا فسيحتوي المتغير BM على نسخة من مدير الخاصية PropertyManager الذي يتحكم بالارتباط بعنصر البيانات.. أما إذا كان الكائن مركبا ويحتوي على قائمة من العناصر، فسيحتوي المتغير BM على نسخة من مدير التسلسل CurrencyManager الذي يتحكم في ربط قائمة عناصر الكائن.

وتمتلك هذه الفئة الخصائص التالية:

### الارتباطات Bindings:

تعيد مجموعة الارتباطات BindingsCollection التي تحتوي على جميع الارتباطات التي تشترك فيها الأداة الحالية.

### العدد Count:

تعيد عدد العناصر المشتركة في الارتباط.. هذا العدد سيكون دائما 1 إذا كان الكائن بسيطا وتم الارتباط بإحدى خصائصه.. أما إذا كان الكائن معقدا وتم الارتباط بقائمة عناصر موجودة داخله، فإن هذه الخاصية تعيد عدد عناصر هذه القائمة (مثل عدد سجلات الجدول إذا كان الارتباط بعمود في أحد جداول مجموعة البيانات).

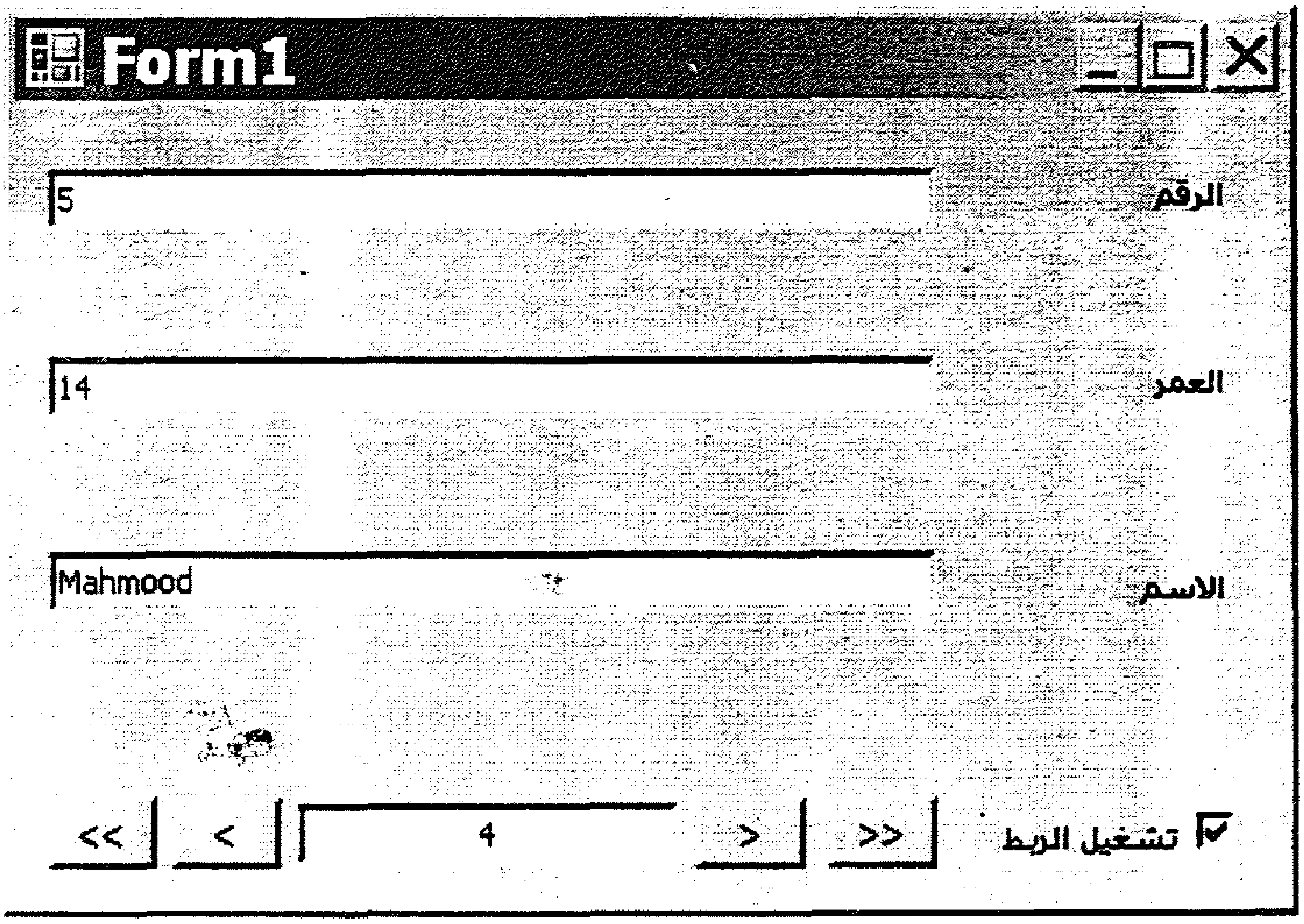
**الموضع Position:**

تقرأ أو تغيّر موضع العنصر الذي تعرضه الأداة حاليًا.. هذا مفيد عند الارتباط بكائن معقد يحتوي على قائمة من العناصر، ففي بدء الارتباط ستوضع في عنصر العرض قيمة أول عنصر في هذه القائمة، ويمكنك بعد هذا أن تستخدم الخاصية Position لعرض أي عنصر آخر في القائمة..

والمثال التالي يعرض في مربع النص ثالث كتاب في جدول الكتب:

```
Dim Bnd As New Binding("Text", Ds, "Books.Book")
TxtBook.DataBindings.Add(Bnd)
Bnd.BindingManagerBase.Position = 2
```

وقد استخدمنا هذه الخاصية في التطبيق BindindSample2 لنتيح للمستخدم التحرك عبر خانات مصفوفة التلاميذ وعرض بيانات كل تلميذ في مربعات النصوص، وذلك باستخدام أزرار الاتجاهات أسفل النموذج:




المريح في الأمر أننا لا نحتاج إلى تغيير الموضع لكل مربع نص على النموذج، فكل ما علينا هو الحصول على مدير الربط الخاص بمصفوفة التلاميذ من خلال محتوى الربط الخاص بالنموذج كالتالي:

```
Dim Bm As BindingManagerBase = Me.BindingContext(Std)
```

وبهذا يؤدي تغيير قيمة الخاصية Bm.Position إلى تغيير العنصر الحالي المعروض في جميع أدوات النموذج.

كما يتيح البرنامج BindindSample2 للمستخدم كتابة رقم الخانة مباشرة في مربع النص الذي يتوسط الأزرار (واسمه TxtPos)، وعندما يضغط Enter من لوحة المفاتيح يتم عرض التلميذ الموجود في هذه الخانة.. لاحظ أن الخاصية Position ترفض أي موضع غير صحيح دون أن يحدث خطأ في البرنامج.. لهذا لو جربت أن تكتب الرقم 10 مثلا في مربع النص وتضغط Enter، فإن الخاصية Position ستنتقل تلقائيا إلى آخر خانة مسموح بها وهي الخانة رقم 4 في هذا المثال.

### الحالي Current:

تعيد الكائن Object المرتبط حاليا بعنصر العرض.. وفي حالة الارتباط بكائن بسيط يحتوي على عدة خصائص (مثل كائن الحجم Size)، تعيد هذه الخاصية هذا الكائن، أما عند الارتباط بكائن معقد يحتوي على قائمة من العناصر، فإن هذه الخاصية تعيد العنصر الحالي في القائمة (الموجود في الموضع الذي تحدده الخاصية Position).

### هل الربط متوقف IsBindingSuspended:

تعيد True إذا كان الربط بين الأداة ومصدر البيانات متوقفا حاليا.

وتمتلك هذه الفئة الوسائل التالية:

### إضافة جديد AddNew:

تضيف عنصرا جديدا إلى قائمة عناصر مصدر البيانات.. وتسبب هذه الوسيلة خطأ في البرنامج إذا كانت الأداة مرتبطة بكائن بسيط لا يحتوي على قائمة داخلية، أو إذا كان الكائن مصفوفة.. وعند استخدام هذه الوسيلة مع صفوف البيانات، تكون حالة السجل الجديد هي RowState.Added، ولن تتم إضافته إلى قاعدة البيانات إلا عند تحديثها.

### حذف من موضع RemoveAt:

تحذف العنصر الموجود في الموضع المرسل كمعامل.. وتسبب هذه الوسيلة خطأ في البرنامج إذا كانت الأداة مرتبطة بكائن بسيط لا يحتوي على قائمة داخلية، أو كان الكائن مصفوفة أو كان لا يمثل الواجهة IBindingList.. وعند استخدام هذه الوسيلة للتعامل لحذف صف بيانات، فسيتم تغيير حالته إلى Deleted، ولن يتم حذفه من قاعدة البيانات إلا عند تحديثها.

**إلغاء التحرير الحالي CancelCurrentEdit:**
تُلغي عمليّة التحرير الحاليّة وتعيد إلى السجل قيمه الأصلية.. وليس لهذه الوسيلة أي تأثير إلا على الكائنات التي تمثل الواجهة IEditableObject مثل فئة سجلّ العرض DataRowView Class.

**إنهاء التحرير الحالي EndCurrentEdit:**
تُنهي عمليّة التحرير مع إبقاء التغييرات التي حدثت للسجل.. وليس لهذه الوسيلة أي تأثير إلا على الكائنات التي تمثل الواجهة IEditableObject مثل فئة سجلّ العرض DataRowView Class.

**معرفة خصائص العنصر GetItemProperties:**
تعيد مجموعة واصفات الخصائص PropertyDescriptorCollection التي تصف خصائص المجموعة المشتركة في الارتباط.. مثلا: عند الارتباط بحقل اسم الكتاب في جدول الكتب، ستحتوي هذه المجموعة على واصف الخصائص PropertyDescriptor لكل عمود في جدول الكتب.. والمثال التالي يعرض اسم أول حقل في السجل الحالي، ويعرض قيمته:

```
Dim BM = Bnd.BindingManagerBase
Dim PD = BM.GetItemProperties(0)
MsgBox(PD.Name) ' ID
MsgBox(PD.GetValue(BM.Current))
```

حيث ستعرض الرسالة الأولى اسم الحقل ID بينما ستعرض الرسالة الثانية قيمة الحقل ID في السجل الحالي.

**إيقاف الارتباط SuspendBinding:**
توقف الارتباط بين الأدوات ومصدر البيانات مؤقتا.. وقد استخدمنا هذه الوسيلة لإيقاف الربط في التطبيق BindingToArray، وذلك عند إزالة علامة الاختيار من مربع الاختيار CheckBox الموجود أسفل النموذج.

**مواصلة الارتباط ResumeBinding:**
تستأنف الارتباط بين الأدوات ومصدر البيانات.

وتمتلك هذه الفئة الأحداث التالية:

- **الربط اكتمل BindingComplete:**

مماثل للحدث Binding.BindingComplete.

- **الموضع تغير PositionChanged:**

ينطلق إذا تغيّرت قيمة الخاصية Position.. والمعامل الثاني e لهذا الحدث من النوع EventArgs، الذي لا يحمل أية معلومات هامة عن الحدث.
وقد استخدمنا هذا الحدث في التطبيق BindingToArray، لتحديث الموضع المعروض في مربع النص TxtPos الموجود أسفل النموذج، كلما تغير الموضع الحالي بسبب ضغط أزرار التحرك.. لاحظ أننا ربطنا هذا الحدث بالإجراء المستجيب له باستخدام الجملة AddHandler في حدث تحميل النموذج كالتالي:

```
AddHandler Bm.PositionChanged,
          AddressOf Bm_PositionChanged
```

- **السجل الحالي تغير CurrentChanged:**

ينطلق إذا تغيّرت قيمة الخاصية Current.

- **العنصر الحالي تغير CurrentItemChanged:**

ينطلق إذا تغيرت حالة العنصر الحالي الذي تعرضه الأداة.. يحدث هذا إذا تغيرت قيمة إحدى خصائص هذا العنصر، أو إذا تم استبداله أو حذفه.

- **خطأ البيانات DataError:**

ينطلق إذا قام مدير الربط بمعالجة خطأ حدث أثناء عملية الربط.. والمعامل الثاني e لهذا الحدث من النوع BindingManagerDataErrorEventArgs، وهو يمتلك الخاصية Exception التي تعيد كائن الاستثناء Exception الذي يحتوي معلومات عن الخطأ الذي حدث.

## فئة مدير الخاصية PropertyManager Class

هذه الفئة ترث الفئة BindingManagerBase، وهي تعمل كمدير يتحكم في ربط كائن بسيط له عدة خصائص ولا يحتوي على قائمة عناصر داخلية.
ولا تمتلك هذه الفئة أية خصائص أو وسائل أو أحداث جديدة غير ما ترثه من الفئة الأم.

## فئة مدير التسلسل CurrencyManager Class

هذه الفئة ترث الفئة BindingManagerBase، وهي تعمل كمدير يتحكم في ربط كائن مركب يحتوي على قائمة عناصر داخلية.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة العنصرين التاليين:

**القائمة List:**

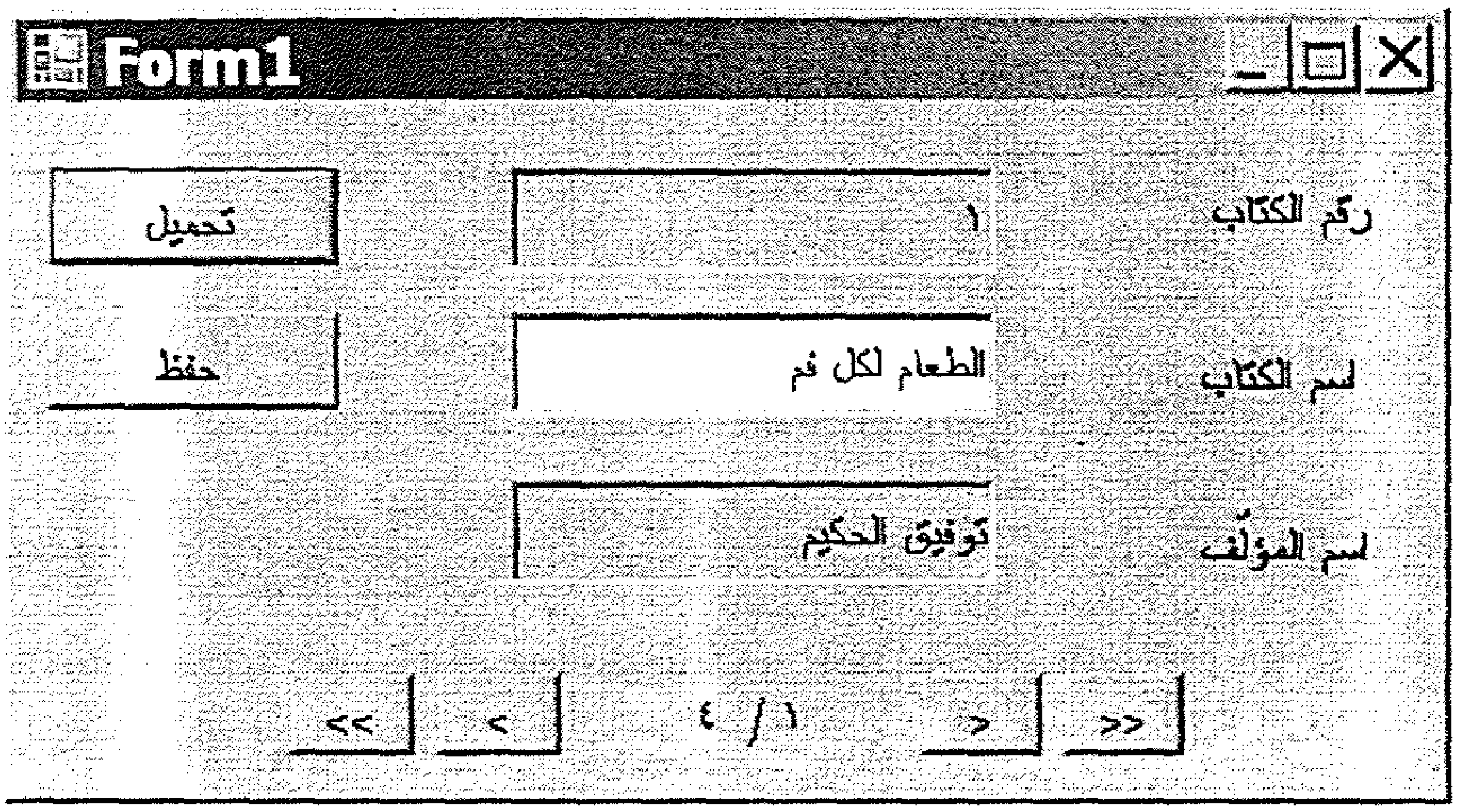
هذه الخاصية من نوع واجهة القائمة IList، هي تعيد القائمة الداخلية التي يحتويها الكائن.

**إنعاش Refresh:**

تعيد ملء مصفوفة العناصر المرتبطة.. استخدم هذه الوسيلة عند الارتباط بكائنات لا تعطي تنبيها عند تغير عناصرها، مثل المصفوفات Arrays.

## ربط الأدوات في وقت التصميم:

يقدم لك مصمم النماذج Form Designer في دوت نت تسهيلات كثيرة لربط الأدوات في وقت التصميم، لتقليل الكود الذي تحتاجه لأداء هذه العملية.. ولكي ترى هذا عمليا، ابدأ مشروعا جديدا وأسمه ViewAndEditBooks، وصمّم واجهة استخدامه كما في الصورة التالية:




أضف إلى النموذج موصل بيانات اسمه DaAuthorBooks، واجعله يستخدم الاستعلام التالي للحصول على أسماء المؤلفين وأسماء كتبهم:

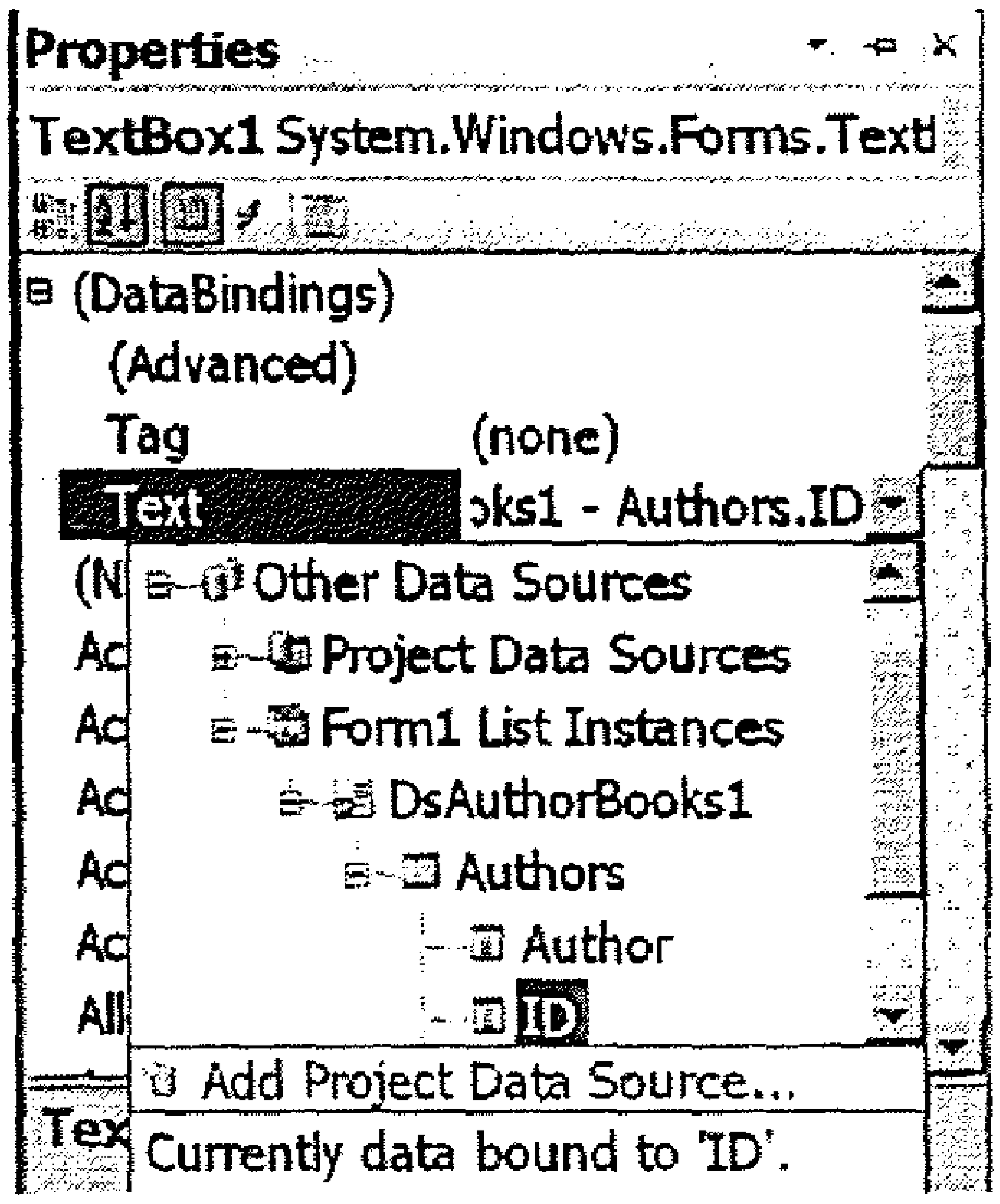
```
SELECT Authors.Author, Books.ID, Books.Book
FROM   Authors INNER JOIN
Books ON Authors.ID = Books.AuthorID
```

أنشئ مجموعة بيانات محددة النوع Typed DataSet من هذا الموصل بالطريقة المعهودة، وأسمها DsAuthorBooks، وأضف نسخة منها إلى النموذج اسمها DsAuthorBooks1.

الآن، نريد ربط مربّعات النصوص بهذه المجموعة.. اتبع الخطوات التالية:

- حدد مربّع النصّ الذي سيعرض رقم الكتاب، ومن نافذة الخصائص حدّد المقطع المسمّى DataBinding.. ستجده في بداية الخصائص، خروجا عن الترتيب الأبجديّ للخصائص، وسيكون موضوعا بين قوسين.
- اضغط العلامة + المجاورة للمقطع DataBinding، لإسدال خصائص الأداة التي يمكن ربطها بمجموعة البيانات.. بالنسبة لمربّع النصّ، ستجد أنّ بإمكانك ربط الخاصيتين Tag و Text.. في الغالب يتمّ ربط الخاصيّة Tag برقم السجلّ ID، وذلك لتسهيل التعامل مع الحقل عند الحاجة.. ولكن في مثالنا هذا، سنعرض هذا الرقم في الخاصيّة Text لتظهر للمستخدم.

- حدّد الخاصيّة Text، واضغط زرّ الإسدال الموجود في خانة القيمة.. ستعرض لك القائمة المنسدلة اختيارين:




1- None: لإلغاء ربط الخاصية بأي مصدر بيانات.

2- Other Data Sources: ولو أسدلت عناصر هذا الفرع، فستجد تحته عنصرين فرعيين:

أ. Project Data Sources: ستجد تحت هذا العنصر مصادر البيانات العامة للمشروع، مثل فئات مجموعات البيانات محددة النوع مثل DsAuthorBooks.. ولو اخترت أيا من هذه الفئات، فسيتم تعريف نسخة منها لاستخدامها في النموذج الحالي.

ب. Form1 List Instances: ستجد تحت هذا الفرع نسخ الأدوات الموضوعة على النموذج الحالي، والتي تصلح للعمل كقوائم ومصادر بيانات.. وفي مشروعنا هذا، ستجد تحته نسخة مجموعة البيانات DsAuthorBooks1.. لو أسدلت عناصر هذه المجموعة، فستجد تحتها أسماء الجداول المعرفة في مجموعة البيانات محددة النوع.. وستجد في مثالنا هذا جدولا واحدا اسمه Authos، وذلك لأن موصل البيانات قد منح الجدول الناتج من استعلام الربط Join Query الاسم الافتراضي Authors.. أسدل حقول هذا الجدول، واختر الحقل ID.

بهذا نكون قد ربطنا الخاصية Text لمربع النص بالحقل ID في مجموعة البيانات.. وبنفس الطريقة يمكنك ربط الخاصية Text لمربع النص الثاني بالحقل Book، ومربع النص الثالث بالحقل Author.

الآن أنهينا ربط أدواتنا بمصدر البيانات، بحيث لو ملأنا مجموعة البيانات بالسجلات، فستظهر قيم حقول السجلّ الحاليّ في مربّعات النصّ، بدون أن نكتب أي كود لفعل هذا.. وكلما تحركنا من سجلّ إلى آخر، يتمّ عرض قيم حقول السجلّ الجديد في الأدوات آليّا.

عند هذه النقطة، لو شغّلت المشروع وضغطت زرّ تحميل البيانات، فستظهر قيم السجلّ الأوّل في مربّعات النصّ، كلّ حقل في مربّعه الذي حدّدناه.

نريد الآن أن نكتب كود الأزرار التي تسمح للمستخدم بالتنقل بين سجلات مجموعة البيانات.. سيكون الأمر بسيطا، فكل ما علينا هو استخدام مدير الربط

BindingManagerBase، والذي يمكننا الحصول عليه من خلال الخاصية BindingContext الخاصة بالنموذج كالتالي:

```
Dim Bm As BindingManagerBase =
    Me.BindingContext(DsAuthorBooks1, "Authors")
```

الآن تستطيع تغيير الموضع كما تريد، باستخدام الخاصيتين Position و Count التابعتين لمدير الربط.. مثلا، في زر الانتقال إلى السجل التالي، استخدمنا الكود:

```
If Bm.Position < Bm.Count - 1 Then
   Bm.Position += 1
   LbPosition.Text = Bm.Position + 1 & " /  " & Bm.Count
End If
```

لاحظ أن محاولة تغيير السجل الحالي قد تؤدي إلى حدوث خطأ في البرنامج، وذلك لأن مدير الربط سيفحص مربعات النصوص، فإن كانت بعض قيمها تغيرت، فسيحاول حفظها في مجموعة البيانات، وسيحدث خطأ إذا كان المستخدم قد أدخل قيمة غير مناسبة لأحد الحقول.. لحل هذه المشكلة، استخدم المقطع Try Catch لمعالجة أي خطأ من هذا النوع، وفي المقطع Catch استخدم الجملة التالية لإلغاء تحرير السجل الحالي (الذي سبب المشكلة):

```
Bm.CancelCurrentEdit( )
```

لاحظ أن هذا الكود سيعيد قيم كل مربعات النصوص إلى ما كانت عليه.. سيكون هذا مستفزا للمستخدم للغاية لو كان عدد مربعات النصوص كبيرا وكان الخطأ ناتجا عن قيمة خاطئة في واحد منها فقط.. لهذا سيكون من الأذكى أن تلغي تحرير مربع النص الذي سبب المشكلة، أو أن تترك القيم الحالية كما هي، وتترك للمستخدم معرفة مربع النص الذي سبب المشكلة من خلال رسالة الخطأ.

وألفت نظرك مجددا إلى أن كل التغييرات التي يجريها المستخدم على مربعات النصوص يتم حفظها في مجموعة البيانات (وليس في قاعدة البيانات)، لهذا على المستخدم ضغط زر الحفظ لإرسال التغييرات من مجموعة البيانات إلى قاعدة البيانات.. هذا الزر يستخدم أمر التحديث Update الخاص بموصل البيانات، لكن هذا يحتاج إلى بعض العمل منا، لأن موصل البيانات لا ينتج أمر التحديث إذا كان أمر التحديد يعيد حقولا من أكثر من جدول، تاركا لك أنت التحكم في الحقول التي تريد تحديثها وكيفية تحديثها.. ونظرا لأننا سنسمح في هذا البرنامج بتحديث الحقل Books.Book فقط، فسنستخدم أمر التحديث التالي:

```
UPDATE Books
SET Book = @Book
WHERE ID = @Original_ID
```

وستجد تعريف هذا الأمر ومعاملاته في حدث تحميل النموذج.

## ربط مربعات القوائم Binding List Boxs:

رأينا حتى الآن كيف نربط الأدوات البسيطة كمربعات النصوص بمصادر البيانات المختلفة.. لكن ماذا لو أردنا ربط أدوات أكثر تعقيدا مثل القائمة ListBox والقائمة المركبة ComboBox وقائمة الاختيار CkeckedListBox؟

لو حاولت استخدام كائن الربط لربط مصدر البيانات بالخاصية Items لهذه الأدوات، فكل ما ستحصل عليه هو رسالة خطأ، تخبرك أنه لا يمكن الارتباط بالخاصية Items لأنها للقراءة فقط!

إذن فما الحل؟

في الحقيقة، هناك طريقة أخرى لربط مربعات القوائم بمصادر البيانات، فهذه الأدوات لا تحتاج إلى المرور من سجل إلى آخر، فهي قادرة على عرض كل السجلات دفعة واحدة، ومن أجل هذا فهي تمتلك خصائص مجهزة لهذا الغرض، وهي:

### مصدر البيانات DataSource:

ضع في هذه الخاصية الكائن الذي تريد الارتباط به.

### عنصر العرض DisplayMember:

تستقبل نصا، يحدد اسم خاصية الكائن التي سيتم عرض قيمته في القائمة.. وفي المشروع BindingListToArray جعلنا قيمة هذه الخاصية "Name"، لهذا تعرض القائمة أسماء الطلاب.

لاحظ أنك لو تركت هذه الخاصية فارغة، فستعرض القائمة النص الذي تعيده الوسيلة ToString الخاصة بكل عنصر من عناصر مصدر البيانات.

### القيمة المحددة SelectedValue:

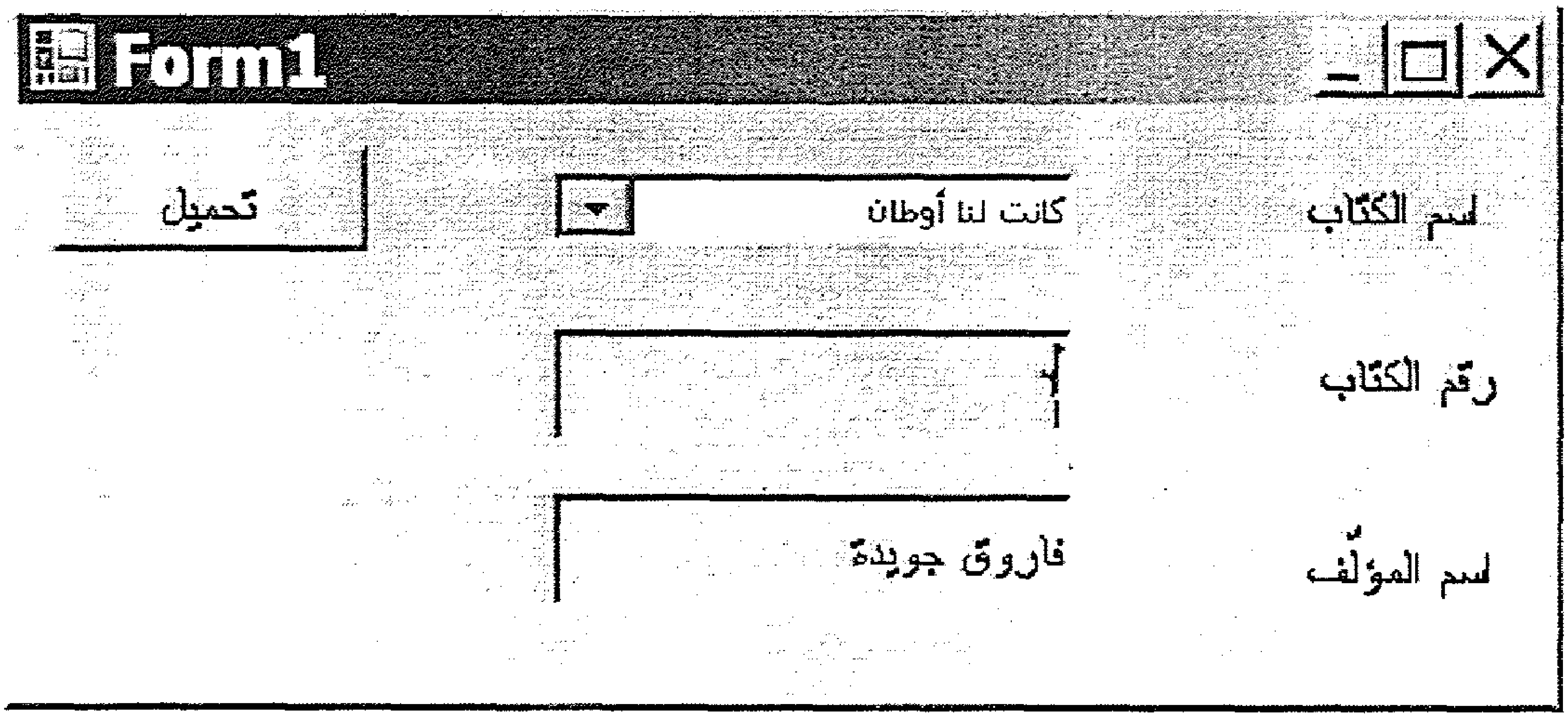
تعيد القيمة المحددة حاليا في القائمة، وهي تتوقف على قيمة الخاصية ValueMember.

### عنصر القيمة ValueMember:

تستقبل نصا، يحدد اسم خاصية الكائن التي ستتم قراءتها عند استخدام الخاصية SelectedValue.. وفي المشروع BindingListToArray جعلنا قيمة هذه الخاصية "Id"، لهذا فإن الخاصية SelectedValue تعيد رقم الطالب المحدد حاليا في القائمة، ويمكنك تجربة هذا بضغط الزر الموجود أسفل القائمة.

ولو تركت الخاصية ValueMember فارغة، فإن الخاصية SelectedValue ستعيد العنصر المحدد في القائمة حاليا مثلها مثل الخاصية SelectedItem.

تعال نستخدم هذه الخصائص في تطوير المشروع ViewAndEditBooks، فهو يبدو عقيما لو حاولت استخدامه لعرض البيانات من قواعد البيانات الضخمة، حيث إنّ التحرّك بين آلاف السجلات واحدا بعد آخر يبدو نوعا من العبث.. لهذا لا بدّ من إنشاء واجهة أكثر ملاءمة لهذا الوضع.. وكحل مبدئي، تعال نستخدم قائمة مركّبة ComboBox لعرض أسماء الكتب، بحيث يختار المستخدم منها اسم الكتاب مباشرة بدلا من ضغط أزرار الانتقال.. صمّم النموذج ليبدو كما في الصورة، وهو موجود في المشروع BookList المرفق بهذا الكتاب:



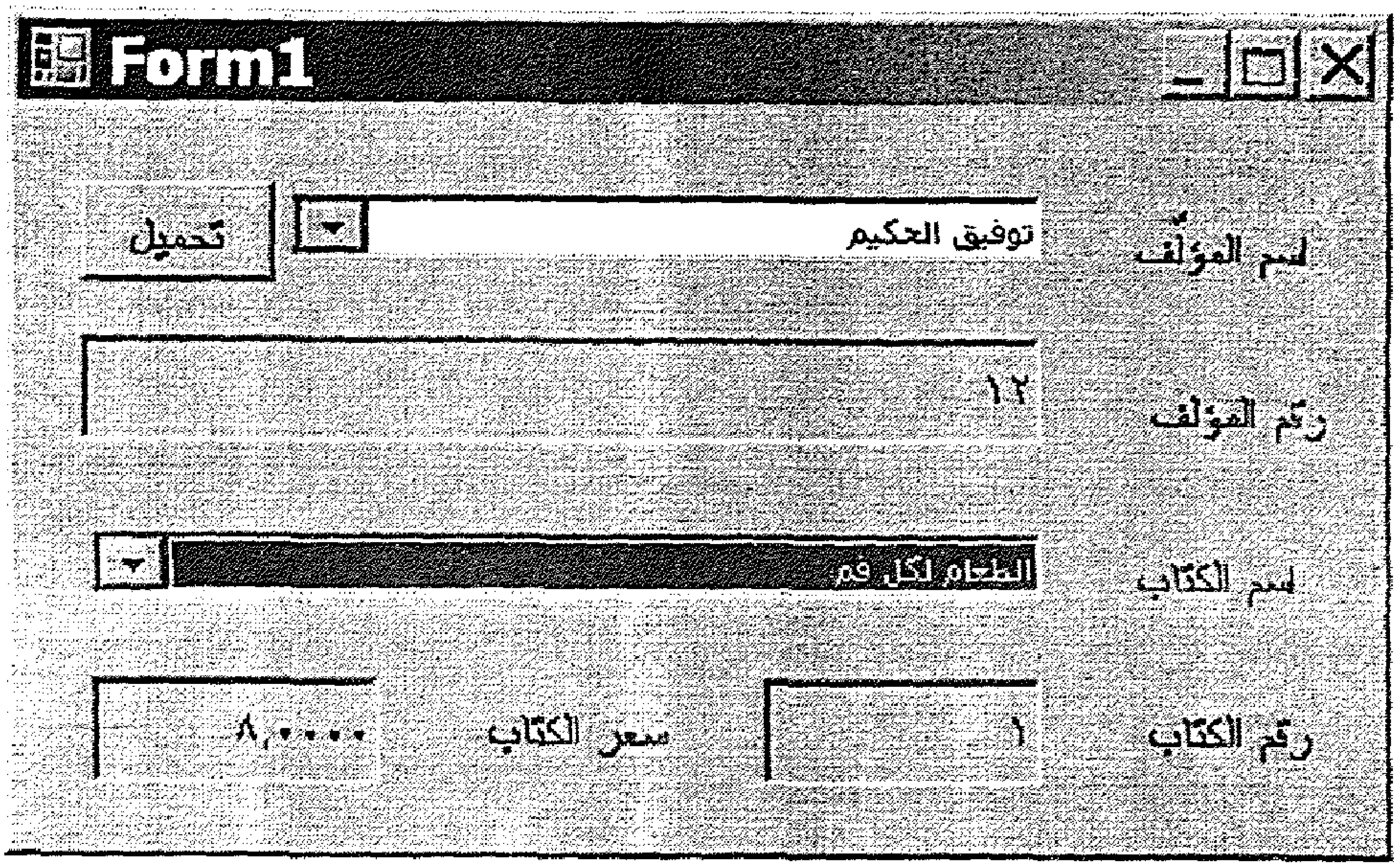

تعرف طبعا كيف تربط مربعي النص اللذين يعرضان اسم المؤلف ورقم الكتاب.. ما يهمنا الآن هو كيفية ربط القائمة المركبة.

حدّد القائمة المركّبة، ومن نافذة الخصائص اختر الخاصيّة DataSource، واضغط زر الإسدال الموجود في خانة قيمتها، ومن القائمة المنسدلة اختر Other Data Sources ثم Form1 List Instances، ثم DsAuthorBooks1.

بعد هذا انتقل إلى الخاصيّة DisplayMember في نافذة الخصائص، واضغط زر الإسدال الموجود في خانة قيمتها، واختر الجدول Authors، ومن حقوله اختر الحقل Book.

إذا شغّلت التطبيق الآن، وضغطت زرّ تحميل البيانات، فستجد أنّ القائمة المركّبة قد امتلأت بأسماء الكتب.. المدهش أنّك لو اخترت اسم أيّ كتاب من القائمة، فسيؤدي هذا إلى تغيير السجل الحالي، ومن ثم سيظهر رقمه واسم مؤلفه في مربّعي النصّ تلقائيا، وبدون أن تكتب سطرا واحدا من الكود لفعل هذا!

حسن.. نريد الآن تطوير المشروع السابق، بحيث نعرض أسماء المؤلفين في قائمة مركبة، ونعرض كتب المؤلف المحدد حاليا في قائمة مركبة أخرى، كما هو موضح في الصورة التالية:




اتبع هذه الخطوات:

1- أنشئ مشروعا جديدا اسمه AuthorsBooks_Lists.
2- أنشأ موصل بيانات اسمه DaAuthors يعيد أسماء المؤلفين وأرقامهم.
3- أنشأ موصل بيانات اسمه DaBooks يعيد أسماء الكتب وأرقامها وأسعارها.. ويعيد أيضا الحقل AuthorID لكي نستخدمه في إنشاء علاقة بين الجدولين.
4- أنشئ مجموعة بيانات محددة النوع اسمها DsAuthorsBooks تحتوي على الجدولين.
5- اضغط الأداة DsAuthorsBooks1 في صينية المكونات بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Edit in Dataset Designer.. ستظهر لك نافذة مخطّط XML.. أنشئ علاقة بين الجدولين اسمها Authors_Books كما فعلنا من قبل.
6- بالنسبة للقائمة المركّبة التي ستعرض أسماء المؤلفين، ومربّع النصّ الذي سيعرض رقم المؤلف، لن تختلف طريقة ربطهما بمجموعة البيانات في شيء عن المثال السابق.
7- أما بالنسبة للقائمة المركّبة التي ستعرض أسماء الكتب فسيختلف الأمر قليلا.. حدّد الخاصيّة DataSource في نافذة الخصائص، وضع فيها القيمة DsAuthorsBooks1.. ثمّ حدّد الخاصيّة DisplayMember، واضغط

زرّ الإسدال الموجود في خانة القيمة.. هذه المرة لا تختر جدول الكتب، بل اختر جدول المؤلفين Authors.. ستجد ضمن عناصره الفرعيّة عنصرا جديدا، هو Authors_Books.. هذا العنصر هو اسم العلاقة التي أنشأناها.. أسدل فروع هذا العنصر.. ستجد تحته أسماء حقول جدول الكتب.. اختر الحقل Book.. بهذا لن تعرض قائمة الكتب كلّ الكتب الموجودة في قاعدة البيانات، بل ستعرض فقط الكتب التي تنتمي إلى المؤلف الحاليّ من خلال العلاقة بينهما.

لاحظ أنك تستطيع أداء هذا من الكود باستخدام الجملة التالية:

```
CmbBook.DisplayMember =
        "Authors.Authors_Books.Book"
```

8- بالنسبة لمربّع النصّ الذي سيعرض رقم الكتاب، اربط الخاصيّة Text بالحقل ID الموجود في العلاقة Authors_Books تحت جدول المؤلفين Authors.. وافعل شيئا مشابها لربط مربع النص الأخير بالحقل Authors.Authors_Books.Price.

لاحظ أنك تستطيع أداء هذا من الكود كما يلي:

```
TextBox1.DataBindings.Add("Text",
    DsAuthorsBooks1, "Authors.Authors_Books.ID")
TextBox2.DataBindings.Add("Text",
    DsAuthorsBooks1, "Authors.Authors_Books.Price")
```

9- وأخيرا، اكتب الكود الذي يملأ مجموعة البيانات بسجلات الجدولين في حدث ضغط زر التحميل:

```
DaAuthors.Fill(DsAuthorsBooks1, "Authors")
DaBooks.Fill(DsAuthorsBooks1, "Books")
```

الآن لو جرّبت البرنامج، فلا ريبَ أنك ستتيه دهشةً وسعادةً، فلديك واجهة استخدام رائعة، تعمل بطريقة مثاليّة، في برنامج لم نكتب فيه أكثر من سطرين من الكود!

لاحظ أننا لا نملك طريقة مباشرة لاستخدام العلاقة Authors_Books بطريقة عكسية في عملية الربط.. مثلا: لا تستطيع أن تجعل مربع نص يعرض مؤلف الكتاب الحالي بالجملة التالية:

```
TextBox1.DataBindings.Add("Text",
        Ds, "Books.Authors_Books.Author")
```

فهذه الجملة ستسبب خطأ في البرنامج، لأن العنصر Authors_Books ليس جزءا من جدول الكتب!.. إن عملية الربط تعتبر العلاقة جزءا من الجدول الرئيسي فقط، وليس الجدول الفرعي!

وقد واجهتنا هذه المشكلة في المشروع BindingTextBox الذي أنشأناه في بداية هذا الفصل، فنحن في هذا المشروع نعرض جدول الكتب في جدول عرض DataGridView، ونريد أن نربط مربع النص باسم مؤلف الكتاب المحدد حاليا في جدول العرض.. في هذه الحالة لا يمكننا أن نستخدم الجملة التالية:

```
TextBox1.DataBindings.Add("Text",
        Ds, "Authors.Authors_Books.Author")
```

لأنها ستعرض في مربع النص اسم أول مؤلف فقط، ولن يتغير مهما تغير الكتاب الحالي.. السبب في هذا أن كائن الربط الخاص بجدول العرض، يتعامل مع سجلات جدول الكتب فقط، وليست له أي علاقة بجدول المؤلفين!

ولحل هذه المشكلة، عرفنا عمودا إضافيا اسمه Author وجعلنا خفيا، وجعلناه يحمل اسم مؤلف الكتاب الحالي من خلال العلاقة بينهما كالتالي:

```
Dim Col As New DataColumn("Author", GetType(String),
    "Parent.Author", MappingType.Hidden)
```

ثم أضفنا هذا العمود إلى جدول الكتب كالتالي:

```
Ds.Books.Columns.Add(Col)
```

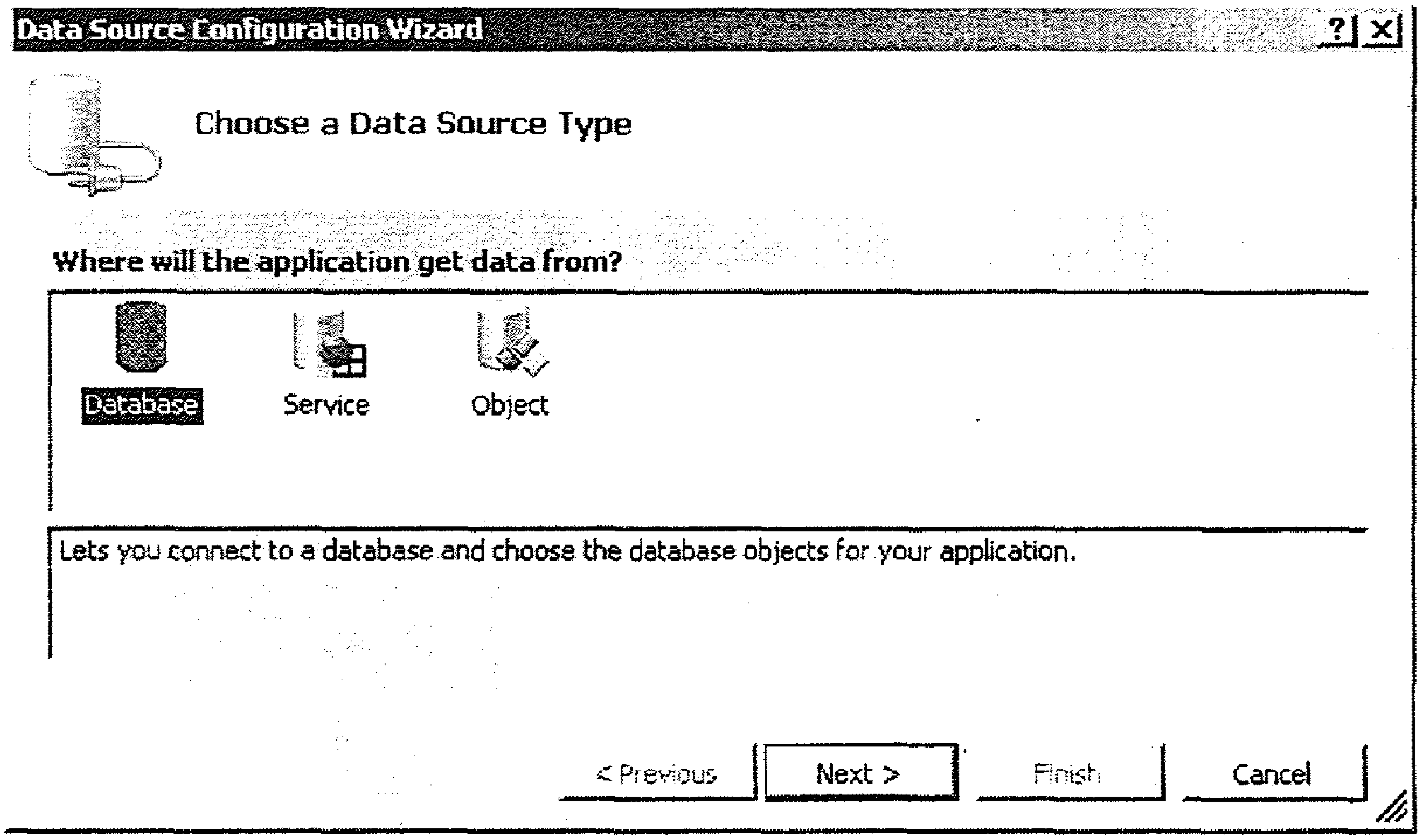
الآن صار من السهل ربط مربع النص بهذا العمود كالتالي:

```
TxtAuthor.DataBindings.Add("Text", Ds, "Books.Author")
```

ولو جربت المشروع فستجده يعمل على ما يرام.

## المعالج السحري لتهيئة مصادر البيانات
## Data Source Configuration Wizard

يستخدم هذا المعالج لإضافة مصادر البيانات إلى مشروعك.. ويمكنك تشغيله بضغط قائمة البيانات Data Menu من شريط القوائم الرئيسية أعلى مصمم النموذج، وضغط الأمر Add New Data Source.. ستظهر لك نافذة اختيار نوع مصدر البيانات، الموضحة في الصورة:




هذه النافذة تتيح لك اختيار أحد أنواع مصادر البيانات التالية:

**1- قاعدة بيانات DataBase:**

يتيح لك هذا النوع إنشاء مصدر بيانات يتعامل مع قاعدة بيانات، حيث يتم إنتاج مجموعة بيانات محددة النوع Typed DataSet وموصلات الجداول اللازمة للتعامل مع كل جدول من جداولها.

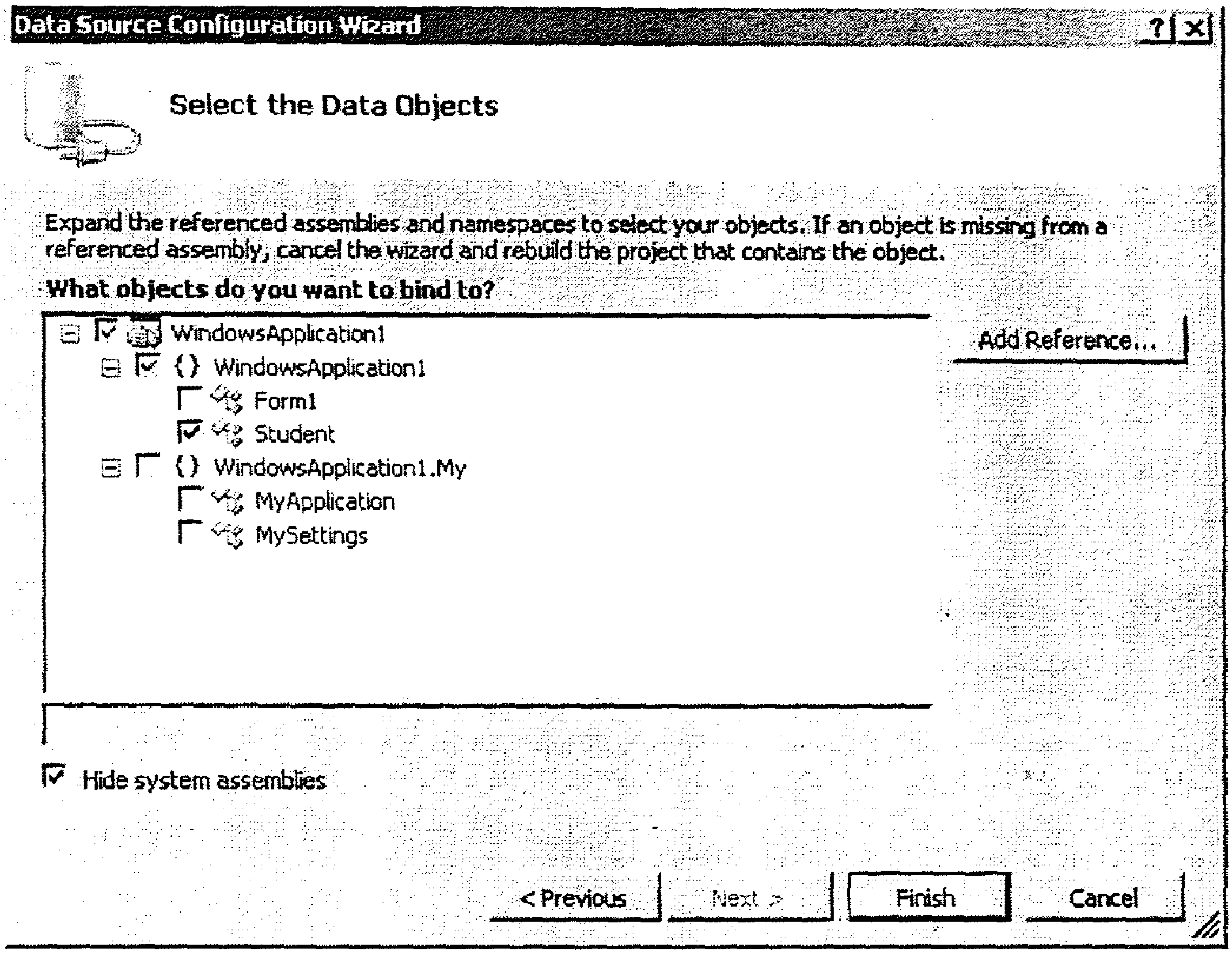
**2- خدمة Service:**

يتيح لك هذا النوع إنشاء مصدر بيانات يتعامل مع خدمة إنترنت Web Service.. هذا النوع خارج نطاق هذا الكتاب.

**3- كائن Object:**

يتيح لك هذا النوع إنشاء مصدر بيانات يتعامل مع أي كائن في مشروعك.. مثلا، لو عرفت فئة اسمها Students، فيمكنك جعلها مصدر

بيانات، باختيار هذا النوع ثم ضغط Next واختيارها من النافذة التاليــة كما هو موضح في الصورة:

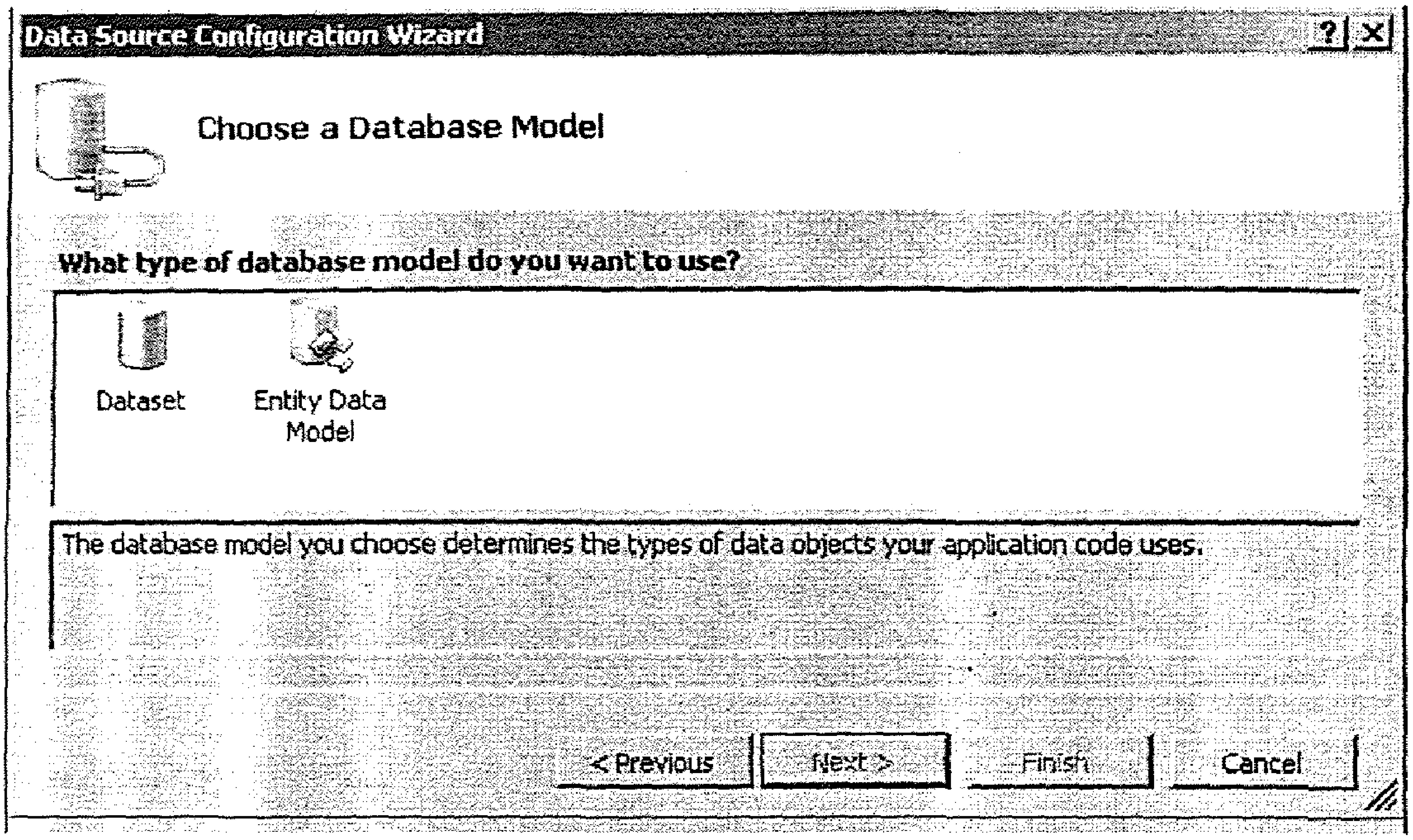



ويمكنك اختيار أي فئة من فئات إطار العمل لاستخدامها كمصدر بيانات لو أردت.. لفعل هــذا أزل علامــة الاختيــار مــن مربــع الاختيــار Hide System Assemblies لتظهر فئات إطار العمل فــي القائمـــة.. وإذا أردت عرض فئــات مــن خـــارج مـــشروعك، فاضـــغط الـــزر Add Reference وأضف مرجعا إلى المكتبات التي توجد بها.. وبعـــد أن تختار مصدر بيانات أو أكثر من القائمة، اضغط الزر Finish.

**4- تطبيق SharePoint:**

يتـــيح لـــك هـــذا النـــوع إنـــشاء مـــصدر بيانـــات يناســـب تطبيـــق SharePoint 2010.. هذا الموضوع خارج نطاق هذا الكتاب.

دعنا الآن نتعامـــل مـــع النـــوع الأكثـــر ملاءمـــة لنـــا هنـــا.. اختـــر النـــوع Database واضغط الزر Next.. ستظهر لك نافذة اختيار نموذج قاعدة البيانات Database Model كما في الصورة:

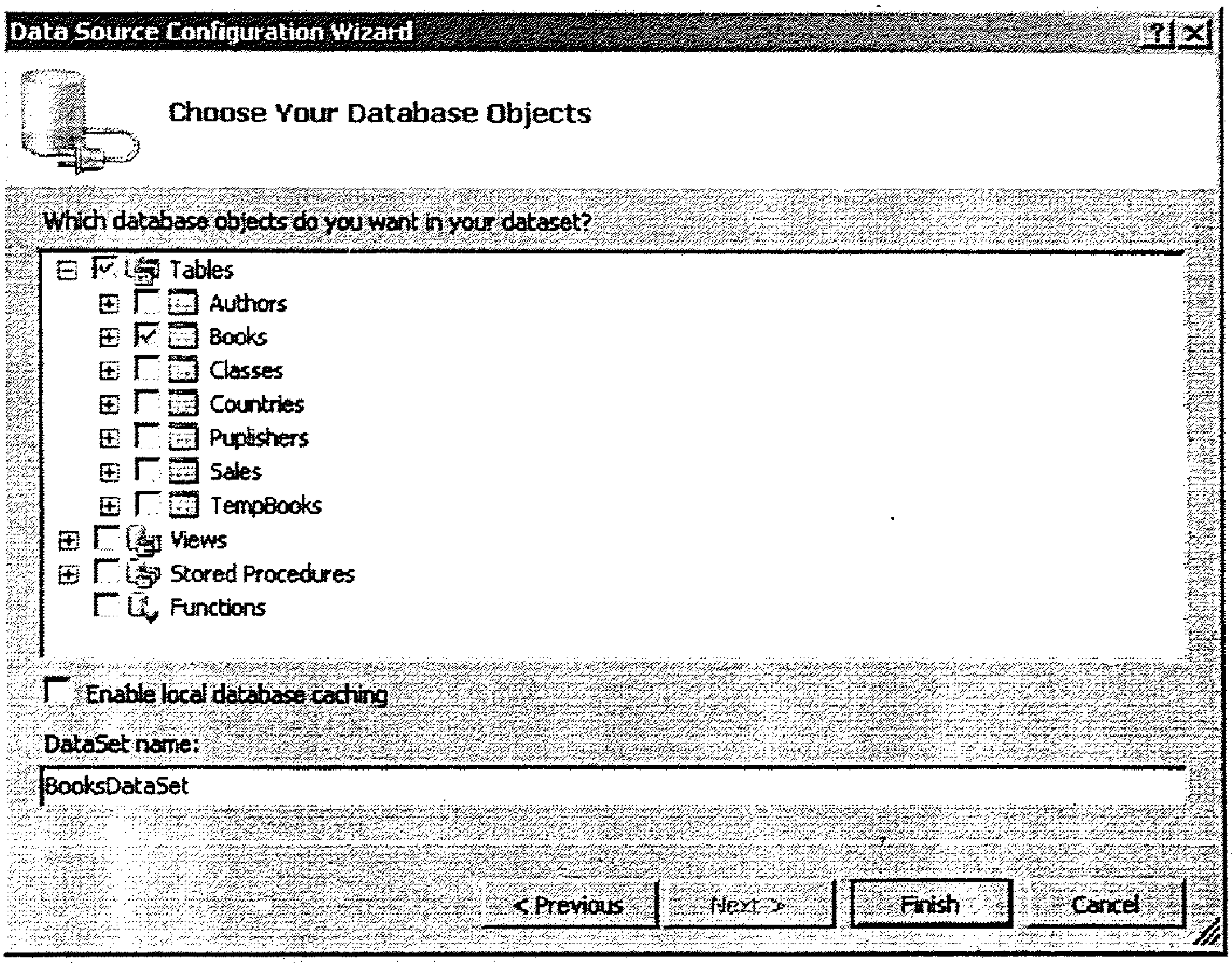



ستجد في هذه النافذة خيارين:

أ- DataSet:
يتم إنشاء مجموعو بيانات محددة النوع، واستخدامها كمصدر بيانات.

ب- Entity Data Model:
هذا الاختيار مناسب للمشاريع التي تستخدم LinQ-To-SQL و Entity Framework، وسنؤجله إلى الكتاب القادم بإذن الله.

اختر DataSet واضغط Next.
ستظهر لك نافذة الاتصال بقاعدة البيانات، وقد تعرفنا عليها كثيرا من قبل.. اختر الاتصال بقاعدة بيانات الكتب Books.mdf، واضغط Next.
ستظهر لك نافذة تسألك إن كنت تريد حفظ نص الاتصال في إعدادات المشروع Settings أم لا.. اترك علامة الاختيار كما هي، وعدل الاسم الذي تريد أن تستخدمه لحفظ نص الاتصال في الإعدادات لو أردت، واضغط Next.
انتظر لحظة إلى أن يتم الاتصال بقاعدة البيانات وتحميل مكوناتها.. ستظهر لك نافذة تتيح اختيار كائنات قاعدة البيانات التي تريد التعامل معها، وستجد فيها كل الجداول والعروض والإجراءات المخزنة المتاحة في قاعدة البيانات، كما تبين الصورة:




اختر جدولي المؤلفين والكتب في مثالنا هذا.. لاحظ أن وضع علامة الاختيار بجوار اسم الجدول يحدد كل أعمدته.. لكنك تستطيع ضغط العلامة + المجاورة لاسم الجدول لإسدال أعمدته، حيث يمكنك إزالة علامة الاختيار المجاورة لبعضها، وبهذا توفر على برنامجك تجميل بيانات لا ضرورة لها.

ويمكنك تعديل الاسم الافتراضي لفئة مجموعة البيانات محددة النوع، من خلال مربع النص السفلي.

تستطيع الآن أن تضغط Finish لإنهاء المعالج السحري وإنشاء مجموعة البيانات، أو يمكنك وضع علامة الاختيار أمام الاختيار:

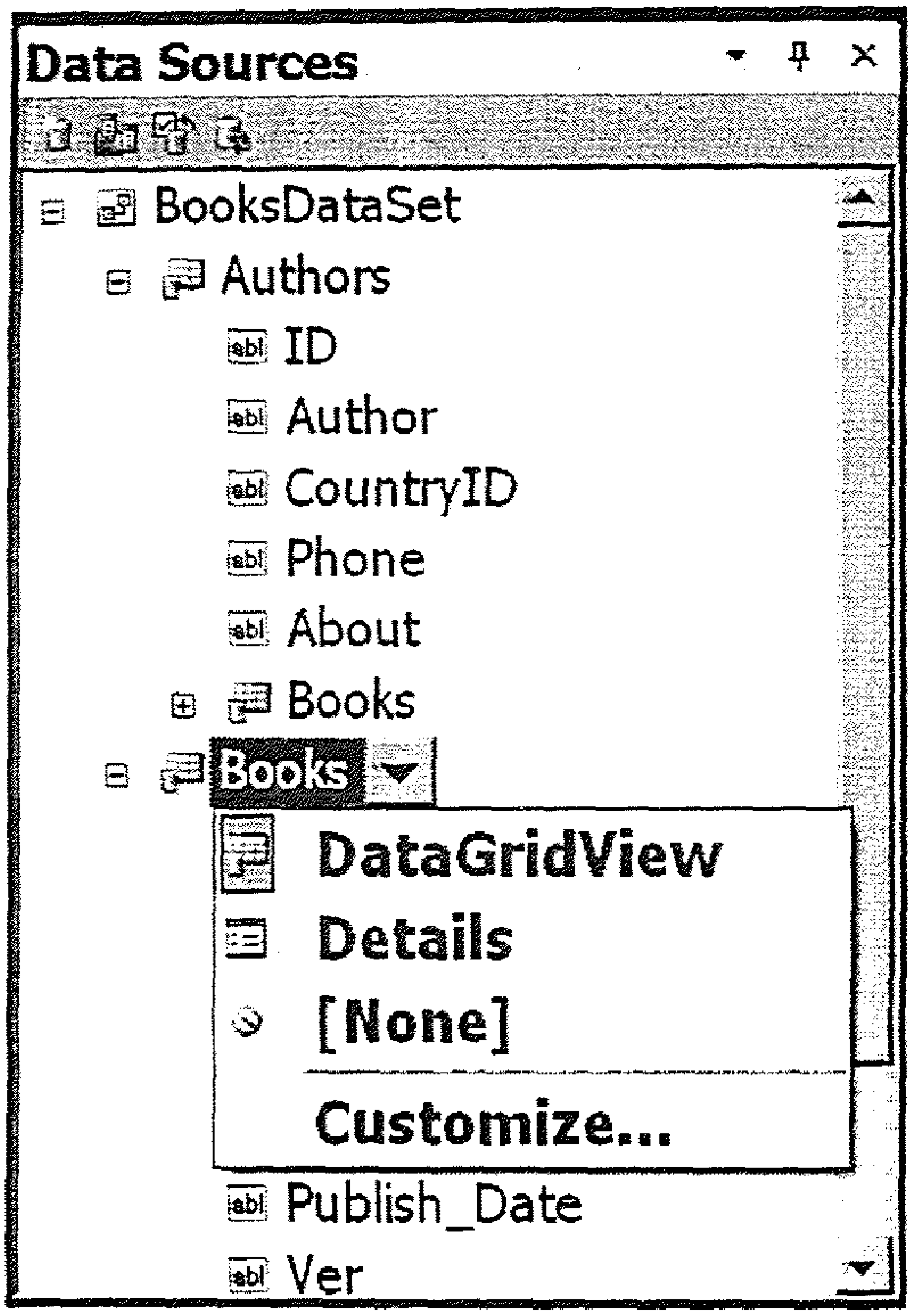
Enable Local Database Caching

هذا الاختيار يتيح حفظ بعض بيانات الجداول على جهاز المستخدم لتكون جاهزة للاستخدام، وذلك إذا كان معدل تغيرها في قاعدة البيانات بطيئا، مما يقلل من عدد مرات الاتصال بالخادم، وبالتالي يحسن أداء وسرعة البرنامج.. إذا اخترت هذا الخيار، فعليك أن تضغط Next لمواصلة المعالج.. لكننا سنترك هذا إلى الكتاب القادم.. اضغط Finish لإنهاء المعالج.

سيؤدي هذا إلى إضافة الملف BooksDataSet.xsd إلى المشروع.. ولو نقرت هذا الملف مرتين، فسيظهر مصمم مجموعة البيانات، الذي سيعرض لك جدول المؤلفين وجدول الكتب والعلاقة بينهما، كما ستجد فيه موصل جدول المؤلفين AuthorDataAdapter وموصل جدول الكتب BookDataAdapter.

## متصفح مصادر البيانات:

لو فتحت القائمة الرئيسية Data وضغطت الأمر Show Data Sources، فسيظهر لك متصفح مصادر البياناتData Sources Explorer كما هو موضح في الصورة.




هذه النافذة تعرض جميع مصادر البيانات الموجودة في المشروع (مثل مجموعات البيانات).. وستجد فيها اسم مجموعة البيانات BoksDataBooks التي أنشأها معالج مصادر البيانات، ولو أسدلت عناصرها، فستجد تحتها جدولي المؤلفين والكتب، ولو أسدلت كلا منهما فستجد تحته أسماء أعمدته.

ويتيح لك متصفح مصادر البيانات إضافة مصادر بيانات جديد وتعديل المصادر الموجودة به، وذلك من خلال الأزرار التي تظهر أعلاه، وهي:

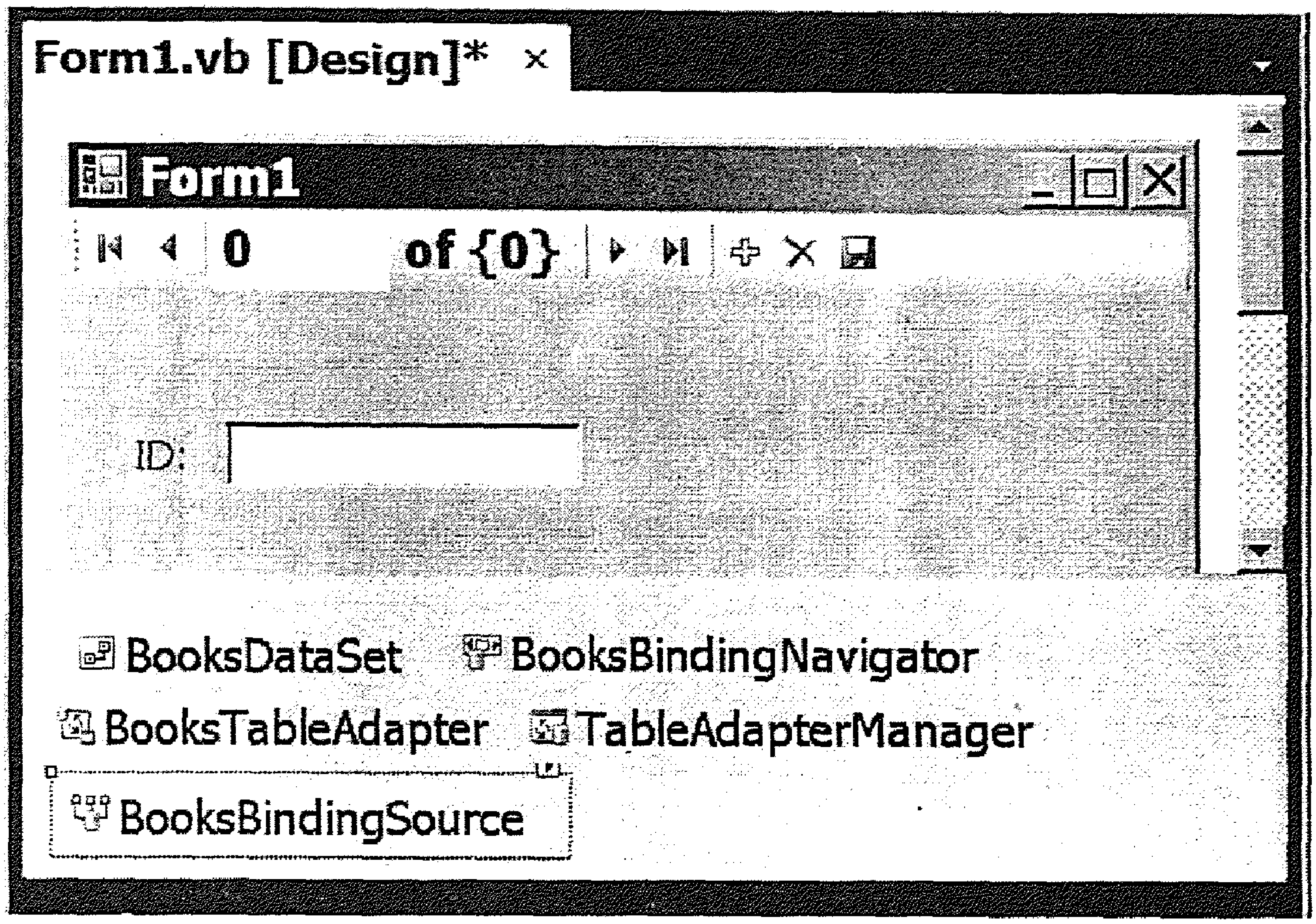
Add New Data Source: يؤدي ضغط هذا الزر إلى تشغيل المعالج السحري لمصادر البيانات.

Edit Data Source With Designer: يؤدي ضغط هذا الزر فتح مصمم مجموعة البيانات لتحريرها.

Configure Data Source With Wizard: يؤدي ضغط هذا الزر إلى تشغيل المعالج السحري لمصادر البيانات، لكنه يعرض نافذة اختيار كائنات قاعدة البيانات مباشرة، لتستطيع إضافة الجداول أو حذفها.

Refresh: يؤدي ضغط هذا الزر إلى إنعاش مصدر البيانات، للتفاعل مع أية تغييرات حدثت في قاعدة البيانات.

ويقدم لك متصفح مصادر البيانات تسهيلات هائلة لتصميم النماذج التي تعرض البيانات، فبمجرد سحب اسم أي حقل من متصفح مصادر البيانات وإلقائه على النموذج، يتم إضافة العديد من الأدوات إلى النموذج كما هو موضح في الصورة:




وكما تلاحظ من الصورة، فإن الأدوات التي أضيفت هي:

1- لافتة تعرض اسم الحقل، اسمها البرمجي XLabel حيث X هو اسم الحقل.
2- مربع نص يعرض قيمة الحقل، اسمه البرمجي XTextBox.
3- نسخة من فئة مجموعة البيانات BooksDataSet لاستخدامها في الحصول على البيانات.
4- نسخة من موصل الجدول الذي يوجد به الحقل.. فمثلا لو سحبت الحقل Book فسيضاف موصل الجدول BooksTableAdapter إلى صينية المكونات.
5- نسخة من مدير التوصيل TableAdapterManager للتحكم تحديث مجموعة البيانات.
6- أداة مصدر الربط BindingSource لاستخدامها في ربط الأدوات بمجموعة البيانات.. وسنتعرف على هذه الأداة بعد قليل.
7- نسخة من الأداة BindingNavigator لتتيح للمستخدم التحرك عبر السجلات.. وسنتعرف على هذه الأداة بعد قليل.

8- يتم إنتاج كود تحميل البيانات في حدث تحميل النموذج Load آليا.. مثلا، يتم إنتاج الكود التالي ليملأ جدول المؤلفين بالبيانات من قاعدة البيانات:

**Me.AuthorsTableAdapter.Fill(Me.BooksDataSet.Authors)**

9- يضاف زر لحفظ التغييرات إلى شريط موجه الربط BindingNavigator، ويضاف الكود التالي إلى حدث ضغط هذا الزر:

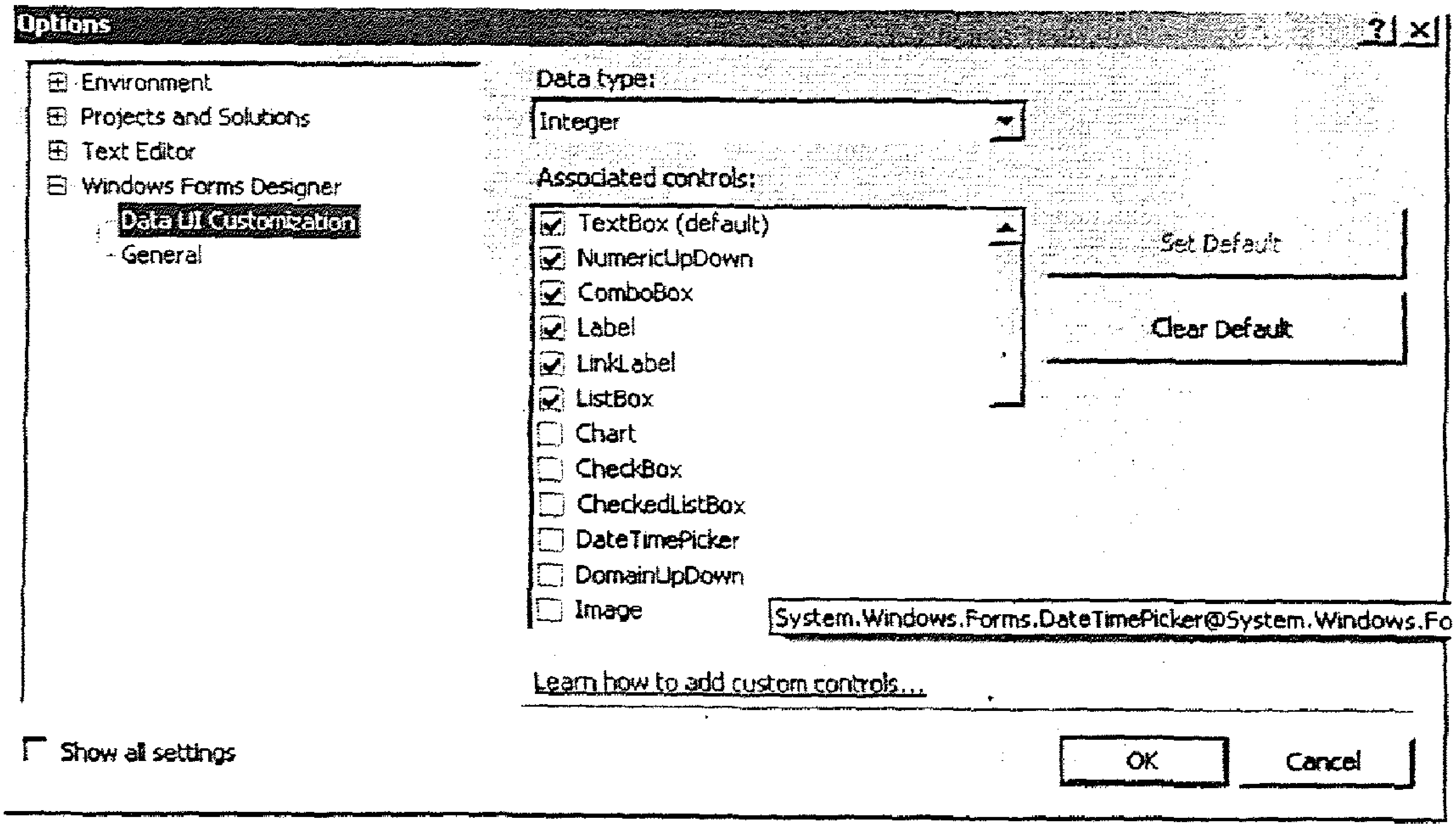
```
Me.Validate()
Me.AuthorsBindingSource.EndEdit()
Me.TableAdapterManager.UpdateAll(Me.BooksDataSet)
```

أليس شيئا رائعا؟.. أنت لا تحتاج إلى فعل أي شيء تقريبا، سوى سحب الحقول وإلقائها على النموذج لتحصل على برنامج كامل الوظيفة!

ويمكنك ربط الحقل بالأداة بطريقة أخرى، وذلك بوضع الأداة على النموذج أولا، ثم سحب الحقل من نافذة المصادر وإلقائه على الأداة.. هذا سيضبط خصائص الأداة تلقائيا لتعرض قيمة هذا الحقل.

كما أنك لست مجبرا على عرض قيمة الحقل في مربع نص، فلو حددت اسم الحقل في متصفح مصادر البيانات، فسيظهر زر إسدال بجواره، ولو ضغطته فستظهر قائمة موضعية، بها أسماء الأدوات التي يمكنك استخدامها لعرض قيمة الحقل.. ولو اخترت القيمة None فلن يتم وضع أدوات لعرض هذا الحقل عند إلقائه على النموذج.

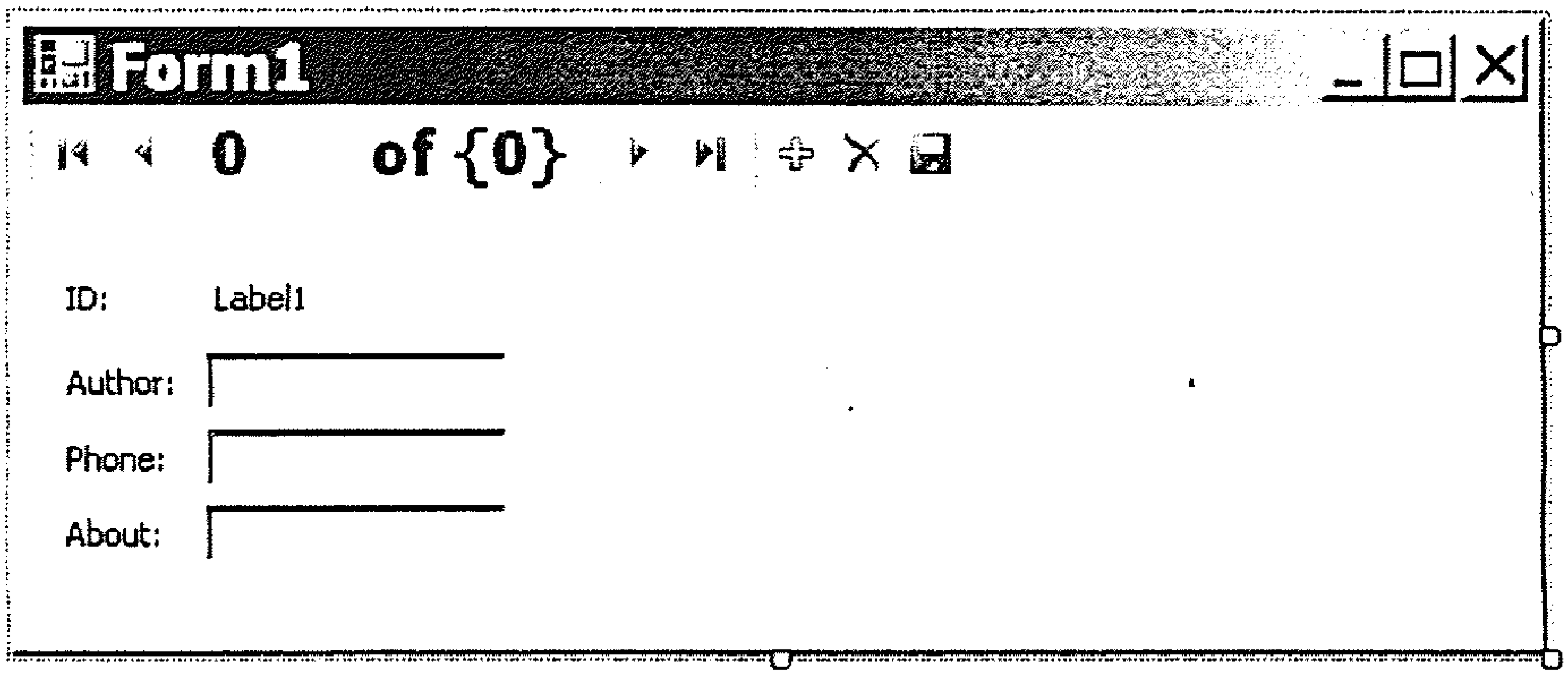
وفي الوضع الافتراضي يكون مربع النص TextBox هو الأداة المستخدمة لعرض قيمة الحقل، لكنك تستطيع اختيار أية أداة أخرى لجعلها تعرض قيمته.. ولو لم تجد الأداة المناسبة بين الأدوات الظاهرة في القائمة، فاضغط الأمر Customize الموجود في نهاية القائمة لعرض النافذة الموضحة في الصورة:



في هذه النافذة يمكنك اختيار نوع البيانات من القائمة المنسدلة Data Type، لتظهر في القائمة السفلية الأدوات التي يمكنها عرض هذا النوع من البيانات، حيث ستجد علامة الاختيار بجوار الأدوات المسموح باستخدامها، ويمكنك وضع علامة الاختيار بجوار أية أدوات أخرى تريد أن تسمح باستخدامها مع هذا النوع، ثم تضغط OK.

وتتيح لك نافذة مصادر البيانات التعامل مع الجدول كله دفعة واحدة.. فلو حددت اسم الجدول Authors في متصفح مصادر البيانات، فسيظهر زر إسدال بجواره، وعند ضغطه ستظهر قائمة موضعية بها الخيارات التالية:

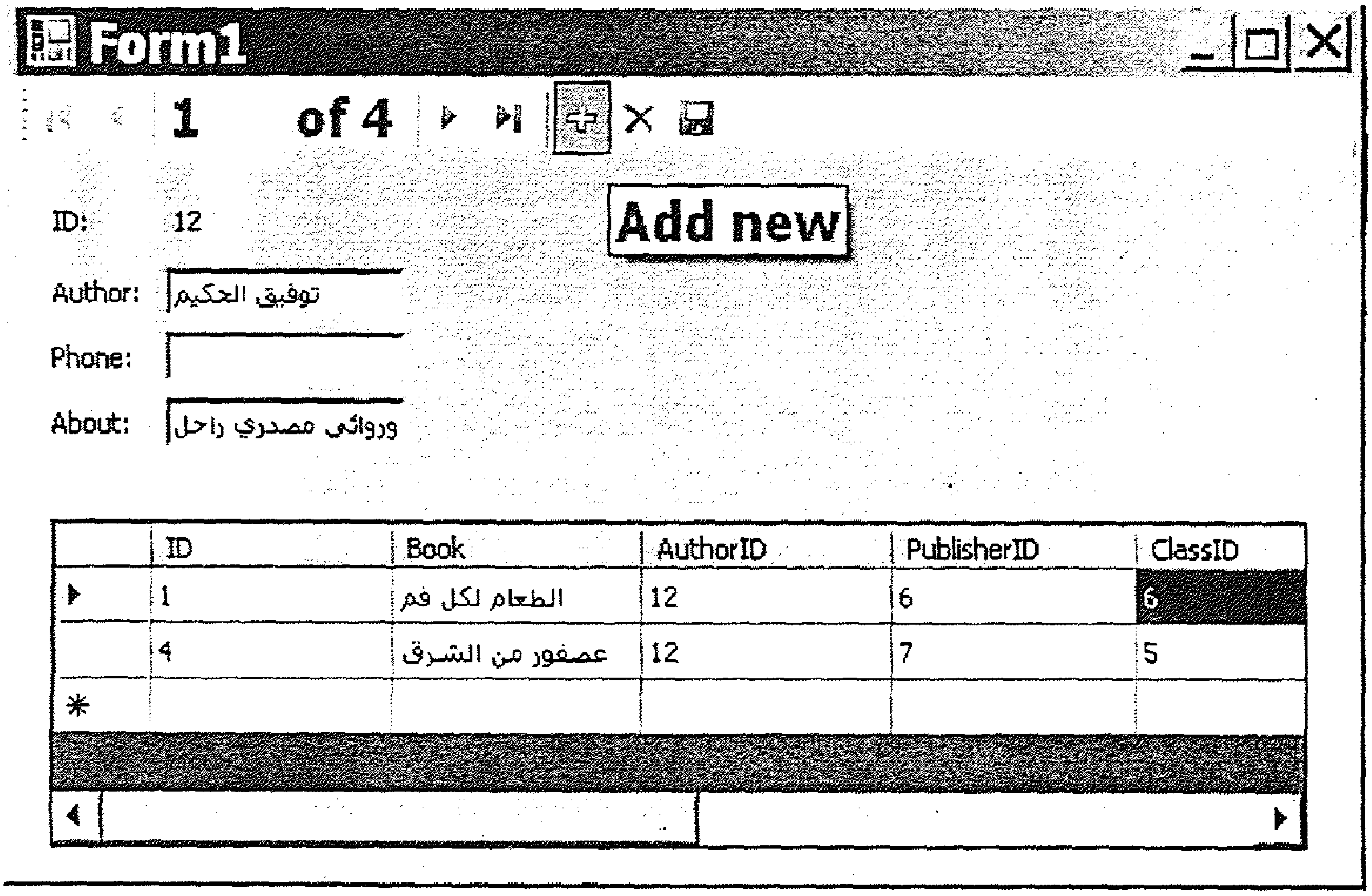
- None: لا يتم وضع أية أدوات على النموذج عند إسقاط الجدول عليه.
- DataGridView: لو اخترت هذا الخيار، وسحبت جدول المؤلفين وألقيته على النموذج، فسيضاف جدول عرض إلى النموذج، وسيحتوي على أعمدة لعرض حقول جدول المؤلفين.. ويؤدي ضغط أزرار التحرك الموجودة على شريط موجه الربط، إلى تغيير السجل المحدد حاليا في جدول العرض، كما أن ضغط زر الحذف سيحذف السجل المحدد حاليا، وضغط زر الإضافة سيضيف سجلا جديدا إلى نهاية جدول العرض.
- Details: لو اخترت هذا الخيار، وسحبت جدول المؤلفين وألقيته على النموذج، فستضاف أداة عرض خاصة بكل حقل على حدة، وبجوارها لافتة تحمل اسم هذا الحقل.. ويختلف نوع أداة العرض الخاصة بكل حقل على حسب الاختيار الذي حددته لكل حقل (كما شرحنا سابقا).. مثلا: قبل أن تسحب جدول المؤلفين، عليك أن تغير نوع أداة عرض الحقل ID إلى لافتة Lable حتى لا تسمح للمستخدم بتغيره، كما يمكنك اختيار العنصر None مع الحقل CountryID لمنع عرض رقم دولة المؤلف.. بعد هذا لو سحبت جدول المؤلفين وألقيته على النموذج، فسيتم وضع الأدوات عليه كما في الصورة:



الأمر رائع فعلا، فقد كان تصميم النماذج في مشاريع قواعد البيانات يستهلك معظم وقت إنتاج البرنامج.. لكن الآن صار الأمر في منتهى البساطة، فأنت تحصل على معظم العمل بالسحب والإسقاط، مع الكثير من الكود المولد آليا!

وبنفس الطريقة يمكنك إضافة نماذج أخرى إلى المشروع، وإسقاط جداول أخرى عليها.

وتمتلك نافذة مصادر البيانات ميزة إضافية هامة، هي السماح لك بعرض البيانات المترابطة.. ولو أسدلت عناصر جدول المؤلفين، فستجد آخر عنصر منها اسمه Books.. هذا العنصر أضيف ليمثل العلاقة المعرفة بين جدول المؤلفين وجدول الكتب في مجموعة البيانات BooksDataSet.. ولو أسدلت العنصر Books، فستجد تحته كل حقول جدول الكتب، ولو سحبتها وألقيتها على النموذج، فستعرض بيانات كتب المؤلف الحالي.. لاحظ أنه من غير المنطقي عرض كل حقل فرعي على حدة إذا كنت تتعامل مع علاقة واحد بمتعدد One-To-Many.. فالمناسب في مثالنا هذا، استخدام جدول لعرض كتب المؤلف الحالي، كما ترى في الصورة:




وستجد هذا التصميم في التطبيق DataSourceWizard المرفق بهذا الكتاب.

لو شغلت هذا التطبيق فسيمكنك الانتقال بين المؤلفين باستخدام شريط موجه الربط، حيث ستعرض الأدوات العلوية بيانات المؤلف الحالي، وسيعرض الجدول السفلي كتب هذا المؤلف.. هذا مشروع Master-Details كامل يعمل بكفاءة دون أن نكتب فيه حرفا واحدا من الكود!.. أليس شيئا مثيرا؟

لاحظ أن وجود العمود AuthorID في جدول العرض لا معنى له.. لكن للأسف، لو حاولت إزالة هذا العمود باختيار None من القائمة المنسدلة للحقل AuthorID قبل سحب عنصر العلاقة Books على النموذج، فلن تنجح.. فجدول العرض يعرض كل الأعمدة شئت أم أبيت، وتكون كل هذه الأعمدة أعمدة مربعات النصوص DataGridViewTextBoxColumn مهما كان نوع الأداة التي اخترتها لعرض قيمة الحقل!.. لهذا عليك تحديد جدول العرض واستخدام نافذة الخصائص لحذف هذا العمود من مجموعة أعمدة جدول العرض Columns Collection.. وسنتعرف على جدول العرض بالتفصيل في الفصل التالي.

ورغم كل التسهيلات التي تمنحها لنا نافذة مصادر البيانات، إلا أنها أحيانا لا تعطينا بالضبط ما نريده.. مثلا: لو أردت عرض أي حقل في قائمة List أو قائمة مركبة ComboBox، فإن سحب الحقل وإلقائه على النموذج يربط الخاصية Text التابعة لهاتين الأداتين بالحقل، ولا يتم ملؤهما بقيم الحقل!

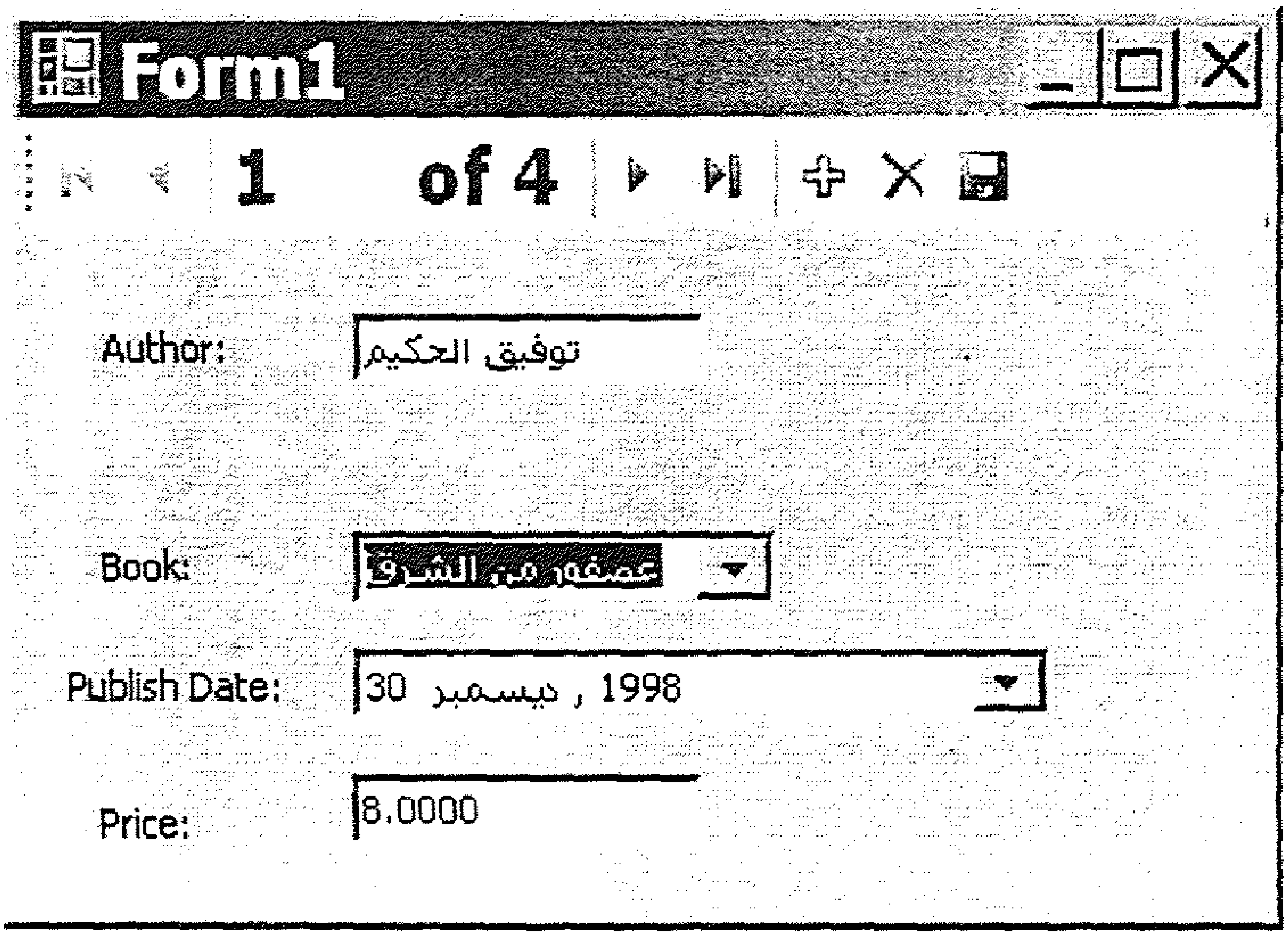
ولحل هذه المشكلة، عليك التدخل يدويا، واستخدام نافذة الخصائص لإزالة الارتباط بالخاصية Text، واستخدام الخاصيتين DataSource و DataMember بدلا منها.

والتطبيق MasterDetails يريك مثالا على هذا.. لإنشاء مثل هذا التطبيق، افعل ما يلي:

- من نافذة مصادر البيانات، اسحب الحقل Authors.Author وألقِه على النموذج لعرض اسم المؤلف الحالي.
- اختر عرض الحقل Authors.Books.Book في قائمة مركبة ComboBox، وألقِه على النموذج ليعرض أسماء الكتب.
- من نافذة الخصائص افتح الخاصية (DataBindings) وأزل الارتباط مع الخاصية Text.
- توجه إلى الخاصية DataSource الخاصة بالقائمة المركبة، واضغط زر الإسدال، ومن القائمة اختر مصدر الربط AuthorsBindingSource وأسدل عناصره الفرعية.. ستجد تحته اسم العلاقة بين جدول المؤلفين وجدول الكتب وهي FK_Books_Authors.. اختر هذه العلاقة كمصدر للبيانات.. سيؤدي هذا إلى إضافة مصدر ربط جديد إلى البرنامج اسمه FKBooksAuthorsBindingSource، وستوضع قيمته تلقائيا في الخاصية DataSource!

- توجه إلى الخاصية DataMember، واضغط زر الإسدال، ومن القائمة اختر الحقل Book.. الآن تأكدنا أن القائمة المركبة ستعرض كتب المؤلف الحالي، لأننا ربطناها من خلال العلاقة بين المؤلفين والكتب.
- من نافذة مصادر البيانات اسحب الحقل Authors.Books.Price وألقه على النموذج.. حدد مربع النص وافتح نافذة الخصائص وافتح الخاصية (DataBindings)، وتوجه إلى الخاصية Text.. اضغط زر الإسدال، واختر العنصر FKBooksAuthorsBindingSource لربط مربع النص من خلال العلاقة.
- يمكنك تكرار هذا مع أكثر من حقل من حقول جدول الكتب.. مثلا، لو سحبت الحقل Publish_Date وألقيته على النموذج، فستظهر أداة اختيار التاريخ والوقت DateDateTimePicker لعرض قيمته.. وأيضا عليك أن تغير ارتباط الخاصية Value الخاصة بهذه الأداة، لتجعلها ترتبط من خلال المصدر FKBooksAuthorsBindingSource كما فعلنا مع مربع النص.

الآن سيكون شكل النموذج كالتالي:




لو شغلت البرنامج، فسيمكنك التحرك عبر المؤلفين باستخدام شريط موجه الربط، حيث ستظهر كتب المؤلف الحالي في القائمة المركبة، وكلما اخترت كتابا منها، يظهر تاريخ نسره في أداة التاريخ، وسعره في مربع النص.. وبهذا نكون حصلنا على طريقة أخرى لعرض البيانات الرئيسية والتفاصيل وتفاصيل التفاصيل!.. صحيح أننا أجرينا بعض التعديلات اليدوية هذه المرة، ولكن صحيح أيضا أننا إلى الآن لم نكتب سطرا واحدا من الكود بأنفسنا!.. مرحى، ما أمتع الكسل!

## واجهة مزود مدير التسلسل
## ICurrencyManagerProvider Interface

تمتلك هذه الواجهة العنصرين التاليين:

- **مدير التسلسل CurrencyManager:**
تعيد مدير التسلسل CurrencyManager التابع لمصدر البيانات الحالي..

- **معرفة مدير التسلسل التابع GetRelatedCurrencyManager:**
أرسل إلى هذه الوسيلة اسم القائمة أو العمود الموجود في مصدر البيانات الحالي، لتعيد إليك مدير التسلسل CurrencyManager الخاص به.. ويمكنك أن ترسل نصا فارغا "" أو Nothing إلى هذه الوسيلة، وفي هذه الحالة ستعيد إليك مدير التسلسل الخاص بمصدر البيانات ككل، وهو نفس مدير التسلسل الذي تحصل عليه من الخاصية CurrencyManager.

## واجهة إلغاء إضافة الجديد
## ICancelAddNew Interface

تضيف هذه الواجهة إلى الفئة التي تمثلها القدرة على قبول العنصر الجديد المضاف أو التراجع عن إضافته، وهي تملك الوسيلتين التاليتين:

- **إلغاء الجديد CancelNew:**
أرسل إلى هذه الوسيلة رقم العنصر الذي أضفته سابقا إلى المجموعة، لتقوم بالتراجع عن إضافة (تقوم بحذفه).

- **إنهاء الجديد EndNew:**
أرسل إلى هذه الوسيلة رقم العنصر الذي أضفته سابقا إلى المجموعة، لتقوم بقبوله نهائيا.. هذا يعني أنك لا تستطيع استخدام الوسيلة CancelNew بعد هذا للتراجع عن إضافة هذا العنصر.

## واجهة إطلاق أحداث التغير
## IRaiseItemChangedEvents Interface

تمتلك هذه الواجهة الخاصية الوحيدة التالية:

**إطلاق أحداث تغير العنصر RaisesItemChangedEvents:**

تعيد هذه الخاصية True، إذا كانت الفئة التي تمثل هذه الواجهة ستطلق الحدث ListChanged إذا حدث تغير في أحد عناصر القائمة الداخلية الخاصة بها.

## فئة قائمة الربط عامة النوع BindingList(Of T) Class

هذه الفئة موجودة في النطاق System.ComponentModel، وهي ترث الفئة Collection(Of T)، وتمثل الواجهات IbindingList و IList و ICancelAddNew و IRaiseItemChangedEvents.

وتعمل هذه الفئة كمجموعة عامة النوع Generic Type Collection، تدعم تقنية ربط البيانات Binding.

ولحدث إنشاء هذه الفئة صيغتان:

1- الصيغة الأولى بدون معاملات.

2- والصيغة الثانية تستقبل قائمة عامة النوع IList(Of T)، لتنسخ عناصرها إلى قائمة الربط.

وإضافة إلى ما ترثه من الفئة الأم، وما تمثله من خصائص ووسائل الواجهات المذكورة، تمتلك هذه الفئة الوسيلتين الجديدتين التاليتين:

**تصفير الارتباطات ResetBindings:**

تطلق الحدث ListChanged مع إرسال القيمة Reset إلى الخاصية e.ListChangedType.

**تصفير العنصر ResetItem:**

تطلق الحدث ListChanged مع إرسال القيمة ItemChanged إلى الخاصية e.ListChangedType.

## واجهة مصدر القائمة
## IListSource Interface

تعمل هذه الواجهة كمصدر للحصول على قائمة List من كائنات لا تمثل واجهة القائمة IList، مما يجعل من الممكن استخدام هذه الكائنات كمصدر للبيانات DataSource عند ربطها بأدوات عرض البيانات.

وتمتلك هذه الواجهة العنصرين التاليين:

**تحتوي على مجموعة قوائم ContainsListCollection:**
تعيد True إذا كانت المجموعة الخاصة بالكائن الحالي تحتوي على قوائم داخلية.

**الحصول على القائمة GetList:**
تعيد مجموعة تمثل واجهة القائمة IList، تحتوي على عناصر الكائن الحالي.

## فئة مصدر الربط BindingSource Class

هذه الفئة ترث الفئة Component لهذا ستجدها في صندوق الأدوات Toolbox تحت الشريط Data، ويمكنك إضافتها إلى صينية مكونات النموذج.
كما تمثل هذه الفئة الواجهات IList و IBindingListView و IcurrencyManagerProvider و ICancelAddNew.
وتحتوي هذه الفئة على قائمة داخلية Internal List تحتوي على عناصر مصدر البيانات، ليتم ربطها بالأدوات الموضوعة على النموذج، وبهذا تسهل هذه الفئة عملية الربط Binding وتتيح لك التحكم فيها كما سنرى بعد قليل.

ولحدث إنشاء هذه الفئة الصيغ التالية:

1- الصيغة الأولى بدون معاملات.
2- الصيغة الثانية لها معامل واحد من نوع الواجهة IContainer، وهو يستقبل الأداة الحاوية التي سينتمي إليها مصدر البيانات، ليتعامل مع الأدوات الموضوعة عليها.. تذكر أن الأدوات الحاوية تشمل النموذج Form واللوحة Panel ومربع التجميع GroupBox... إلخ.
3- الصيغة الثالثة تستقبل الكائن Object الذي يعمل كمصدر للبيانات، ونصا يمثل اسم عنصر البيانات.

وإضافة إلى ما تمثله من خصائص الواجهات المذكورة، تمتلك هذه الفئة الخصائص التالية:

### مصدر البيانات DataSource:

تستقبل هذه الخاصية الكائن Object الذي يعمل كمصدر البيانات.. هذا سيؤدي إلى ما يلي:

- إذا كان الكائن من النوع T، فإن نوع عناصر القائمة الداخلية للفئة BindingSource سيحدد على أنه من النوع T، ولن تقبل هذه القائمة أي بيانات لا يمكن تحويلها إلى هذا النوع.
- إذا وضعت في هذه الخاصية القيمة Nothing، فستظل القائمة الداخلية غير محددة النوع، وستأخذ نوع أول عنصر تضيفه إليها باستخدام الوسيلة Add.. لاحظ أن خطأ سيحدث في البرنامج إذا وضعت قيمة في الخاصية DataMember بينما للخاصية DataSource القيمة Nothing.

- إذا كان الكائن بسيطا لا يحتوي على قائمة من العناصر، فإن القائمة الداخلية ستكون فارغة.
- إذا كان الكائن الذي وضعته في هذه الخاصية مصفوفة Array أو مجموعة Collection، فإن عناصرها ستوضع في القائمة الداخلية.
- إذا كان الكائن معقدا ويحتوي على قائمة من العناصر أو أكثر من قائمة، فيجل عليك ذكر اسم القائمة في الخاصية DataMember (كاسم الجدول في مجموعة البيانات مثلا)، حيث ستوضع عناصر هذه القائمة في القائمة الداخلية.

ويمكنك أن تضع في هذه الخاصية نوع أحد الكائنات بدلا من أن تضع الكائن نفسه.. فبدلا من أن تضع في هذه الخاصية مجموعة بيانات كالتالي:

**Bs.DataSource = Ds**

يمكنك أن تضع نوع مجموعة البيانات كالتالي:

**Bs.DataSource = Ds.GetType( )**

وإذا كانت لديك مجموعة بيانات محددة النوع Typed DataSet اسمها BooksDs فيمكنك استخدام نوعها كمصدر بيانات كالتالي:

**Bs.DataSource = GetType(BooksDs)**

ولكن، فيم يفيدنا هذا؟

في بعض الأحيان تحتاج إلى تصميم بعض أدوات عرض البيانات في وقت التصميم (مثل جدول عرض البيانات DataGridView)، وهذا معناه أنك تحتاج إلى عرض أعمدة الجداول المرتبطة في هذه الأداة.. لكن في وقت التصميم قد لا تكون هناك كائنات معرفة من الفئات التي تعمل كمصادر للبيانات، لهذا تسمح لك هذه الخاصية بوضع نوع هذه الفئات فيها، لتستنتج منه طريقة العرض المطلوبة.

ويمكنك وضع قيمة هذه الخاصية بطريقة مرئية في وقت التصميم، وذلك باستخدام نافذة الخصائص، حيث سيعرض لك زر الإسدال شجرة العناصر المتاحة.. في هذه الشجرة ستجد عنصرين رئيسيين:

1- None: وهي القيمة الافتراضية، وهي تجعل لهذه الخاصية القيمة Nothing.

2- Other Data Sources: وتحتها الاختياران التاليان:

أ. Project Data Sources: ويوجد تحتها كل فئات مصادر البيانات المتاحة في المشروع كله.. ويؤدي اختيار أي فئة من هذه الفئات، إلى إنشاء نسخة جديدة منها وإضافتها إلى النموذج.

ب. Form Data Sources: ويوجد تحتها كل الكائنات المعرفة في النموذج الحالي وتصلح كمصادر بيانات، مثل القوائم Lists ومجموعات البيانات DataSets وغيرها.

وفي الهامش السفلي للنافذة المسدلة، يوجد رابط اسمه:

Add Project data Source

عند الضغط عليه يتم تشغيل المعالج السحري لتهيئة مصادر البيانات Data Source Configuration Wizard، ليمكنك إنشاء مصدر بيانات جديد وإضافته تحت الفرع Project Data Sources.

### عنصر البيانات DataMember:

تستقبل اسم الخاصية أو اسم القائمة أو العمود الموجود في مصدر البيانات، والذي يتم أخذ البيانات منه.

### القائمة List:

تعيد نسخة من الواجهة IList تحتوي على القائمة الداخلية التي تحتوي على العناصر المرتبطة.. لاحظ أن نوع القائمة العائدة يتحدد تبعا لما يلي:

| قيمة الخاصية DataSource | قيمة الخاصية DataMember | القائمة العائدة من الخاصية List |
|---|---|---|
| Nothing | نص فارغ | مصفوفة قائمة ArratList فارغة. |
| Nothing | أي قيمة | عند قراءة الخاصية List سيحدث خطأ في البرنامج. |
| مصفوفة | | مصفوفة Array. |
| كائن يمثل الواجهة IListSource | | القيمة العائدة من الوسيلة IListSource.GetList |
| كائن يمثل الواجهة IBindingList | | نسخة من الواجهة IBindingList. |
| كائن يمثل الواجهة IList | | نسخة من الواجهة IList. |
| كائن بسيط من النوع T لا يحتوي على قائمة | | نسخة من الواجهة IBindingList(Of T) بها عنصر واحد. |
| كائن يمثل الواجهة ICustomType Descriptor | | مصفوفة قائمة ArrayList بها عنصر واحد. |

| قيمة الخاصية DataSource | قيمة الخاصية DataMember | القائمة العائدة من الخاصية List |
|---|---|---|
| كائن يمثل الواجهة IEnumerable | | مصفوفة قائمة ArratList نسخت إليها عناصر الكائن. |
| نوع المصفوفات Array Type | عنصر من النوع T | نسخة فارغة من الفئة BindingList(Of T). |
| نوع كائن يمثل الواجهة IListSource أو الواجهة ITypedList | | نسخة جيدة فارغة من نوع هذا الكائن. |
| نوع كائن يمثل الواجهة IList | عنصر من النوع T | نسخة فارغة من الفئة BindingList(Of T). |
| نوع كائن بسيط لا يحتوي على قائمة | | نسخة فارغة من الفئة BindingList(Of T). |
| نوع كائن يمثل الواجهة ICustomType Descriptor | | عند قراءة الخاصية List سيحدث خطأ في البرنامج. |

**الموضع Position:**

تقرأ أو تغير موضع العنصر الحالي في القائمة الداخلية لمصدر البيانات، وهو العنصر الذي يتم عرضه في الأدوات المرتبطة بمصدر الربط.

**الحالي Current:**

تعيد كائنا Object يحتوي على العنصر الحالي في القائمة الداخلية، وهو العنصر الموجود في الموضع المحدد في الخاصية Position.

**الترتيب Sort:**

تحدد طريقة ترتيب العناصر في القائمة، وهي تستقبل نصا يحتوي على اسم العمود المستخدم في الترتيب، متبوعا باتجاه الترتيب (ASC أو DESC).

**هل الربط متوقف IsBindingSuspended:**

تعيد True إذا كان الربط متوقفا حاليا.

**إطلاق أحداث تغير القائمة RaiseListChangedEvents:**

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسينطلق الحدث BindingSource.ListChanged عندما يحدث تغيير في عناصر القائمة الداخلية.

و تمتلك هذه الفئة الوسائل التالية:

**إلغاء التحرير CancelEdit:**

تنهي عملية تحرير العنصر الحالي، وتلغي أي تغيير حدث له.

**إنهاء التحرير EndEdit:**

تنهي عملية تحرير العنصر الحالي، وتبقي على التغييرات التي حدثت له.

**التحرك إلى الأول MoveFirst:**

تجعل أول عنصر في القائمة الداخلية هو العنصر الحالي (Position = 0).

**التحرك إلى الأخير MoveLast:**

تجعل آخر عنصر في القائمة الداخلية هو العنصر الحالي (Position = Count -1).

**التحرك إلى التالي MoveNext:**

تجعل العنصر التالي في القائمة الداخلية هو العنصر الحالي (Position += 1).

**التحرك إلى السابق MovePrevious:**

تجعل العنصر السابق في القائمة الداخلية هو العنصر الحالي (Position -= 1).

### إزالة الحالي RemoveCurrent:

تزيل العنصر الحالي من القائمة الداخلية.. لاحظ أن خطأ سيحدث في الحالات التالية:

- لو كانت للخاصية BindingSource.AllowRemove القيمة False.
- لو كانت القائمة الداخلية للقراءة فقط Read Only أو ثابتة الحجم Fixed Size.
- لو كان الموضع الحالي غير مقبول، سواء كان أصغر من صفر أو كان أكبر من أو يساوي عدد عناصر القائمة.

### تصفير العنصر الحالي ResetCurrentItem:

تطلق الحدث ListChanged لتطلب من الأدوات التي تعرض العنصر الحالي أن تنعش القيم التي تعرضها.

### تصفير الارتباطات ResetBindings:

تطلق الحدث ListChanged لتطلب من كل الأدوات المرتبطة بمصدر البيانات أن تنعش القيم التي تعرضها، وهي تستقبل معاملا منطقيا، إذا جعلت قيمته True فهذا معناه أن هناك تغييرا في مخطط مصدر البيانات نفسه (كحدوث تغيير في أعمدة الجدول)، وإذا جعلته False فهذا معناه أن التغيير قد حدث في بعض عناصر القائمة الداخلية فقط.

ويتم استدعاء هذه الوسيلة آليا عند تغير قيمة الخاصية DataSource أو الخاصية DataMember، كما يمكنك أن تستدعيها بنفسك إذا قمت بتغيير بعض العناصر في مصدر البيانات.

ولكن.. لماذا نحتاج إلى استخدام الوسيلة ResetBindings لإنعاش كل العناصر، بينما يكفينا إنعاش العنصر الحالي باستخدام الوسيلة ResetCurrentItem؟.. ألا تعرض الأدوات العنصر الحالي فقط؟

والإجابة هي أن بعض الأدوات تعرض أكثر من عنصر في نفس الوقت (كالقائمة ListBox وجدول عرض البيانات DataGridView)، بينما بعض الأدوات تعرض السجل الحالي فقط (مثل مربع النص واللافتة).. لهذا إذا حدث تغيير في عدد من العناصر وكنت تعرض البيانات في قائمة أو جدول عرض، فعليك باستدعاء الوسيلة ResetBindings، أما إذا كنت تستخدم أدوات عرض بسيطة كمربع النص واللافتة وحدث تغير في العنصر الحالي، فاستخدم الوسيلة ResetCurrentItem.. أما إذا كان التغير في عنصر غير العنصر الحالي، فلا تحتاج إلى إنعاش الأدوات البسيطة، لأنها ستنعش نفسها تلقائيا عند الانتقال إلى العنصر الذي تغير.

- **تصفير العنصر ResetItem:**

تطلق الحدث ListChanged لتطلب من الأدوات التي تعرض العنصر الذي تغير في السجل الحالي بأن تنعش القيمة التي تعرضها.. وتستقبل هذه الوسيلة رقم العنصر المراد إنعاشه.

ويتم استدعاء هذه الوسيلة آليا كلما حدث تغيير لأحد عناصر القائمة الداخلية، لكن بإمكانك استدعاؤها بنفسك أيضا.

- **إيقاف الربط SuspendBinding:**

توقف ربط المصدر الحالي بالأدوات مؤقتا.

- **مواصلة الربط ResumeBinding:**

تعيد ربط الأدوات بالمصدر الحالي.

وإضافة إلى ما تمثله من أحداث الواجهات التي تمثلها، تمتلك هذه الفئة الأحداث التالية، وكلها مألوف لنا لهذا لن نكرر شرحها هنا:

- **الربط اكتمل BindingComplete**
- **الحالي تغير CurrentChanged**
- **العنصر الحالي تغير CurrentItemChanged**
- **عنصر البيانات تغير DataMemberChanged**
- **مصدر البيانات تغير DataSourceChanged**
- **الموضع تغير PositionChanged**
- **خطأ البيانات DataError**

## فئة مساعد ربط القوائم ListBindingHelper Class

تحتوي هذه الفئة على بعض الوسائل المشتركة Shared Methods، التي تستخدمها الفئة BindingSource في التعامل مع مصدر البيانات.. وهذه الوسائل هي:

### معرفة القائمة GetList:

تستقبل الكائن الذي يعمل كمصدر بيانات، وتعيد قائمة البيانات التي يحتويها، والتي يمكن الارتباط بها إن وجدت، فإن لم توجد، فإن هذه الوسيلة تعيد كائن مصدر البيانات نفسه.

وتوجد صيغة أخرى لهذه الوسيلة، لها معامل ثان، يستقبل اسم الخاصية التي تعمل كعنصر البيانات في مصدر البيانات، للحصول على قائمة العناصر الخاصة بها.

### معرفة اسم القائمة GetListName:

تعيد اسم القائمة إن وجدت، أو اسم نوع مصدر البيانات، ولها معاملان:

- الكائن الذي يعمل كمصدر للبيانات.
- مصفوفة من واصفات الخصائص PropertyDescriptor، التي تحدد القائمة المراد معرفة اسمها.

### معرفة نوع عناصر القائمة GetListItemType:

هذه الوسيلة مماثلة في صيغتيها للوسيلة السابقة، إلا أنها تعيد كائن النوع Type، الذي يمثل نوع عناصر القائمة.

### معرفة خصائص عناصر القائمة GetListItemProperties:

تعيد مجموعة واصفات الخصائص PropertyDescriptorCollection، التي تصف خصائص عنصر القائمة.. ولهذه الوسيلة ثلاثة معاملات:

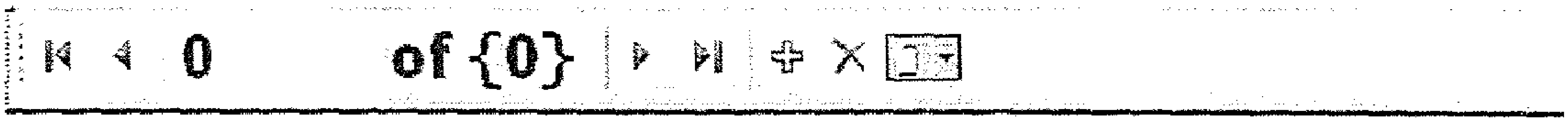
- الكائن الذي يعمل كمصدر للبيانات.
- اسم عنصر البيانات.
- مصفوفة من واصفات الخصائص PropertyDescriptor، التي تحدد القائمة المراد التعامل معها.

وتوجد صيغتان أخريان لهذه الوسيلة، إحداهما تستقبل المعامل الأول فقط، والأخرى تستقبل المعاملين الأول والثالث فقط.

## فئة موجه الربط BindingNavigator Class

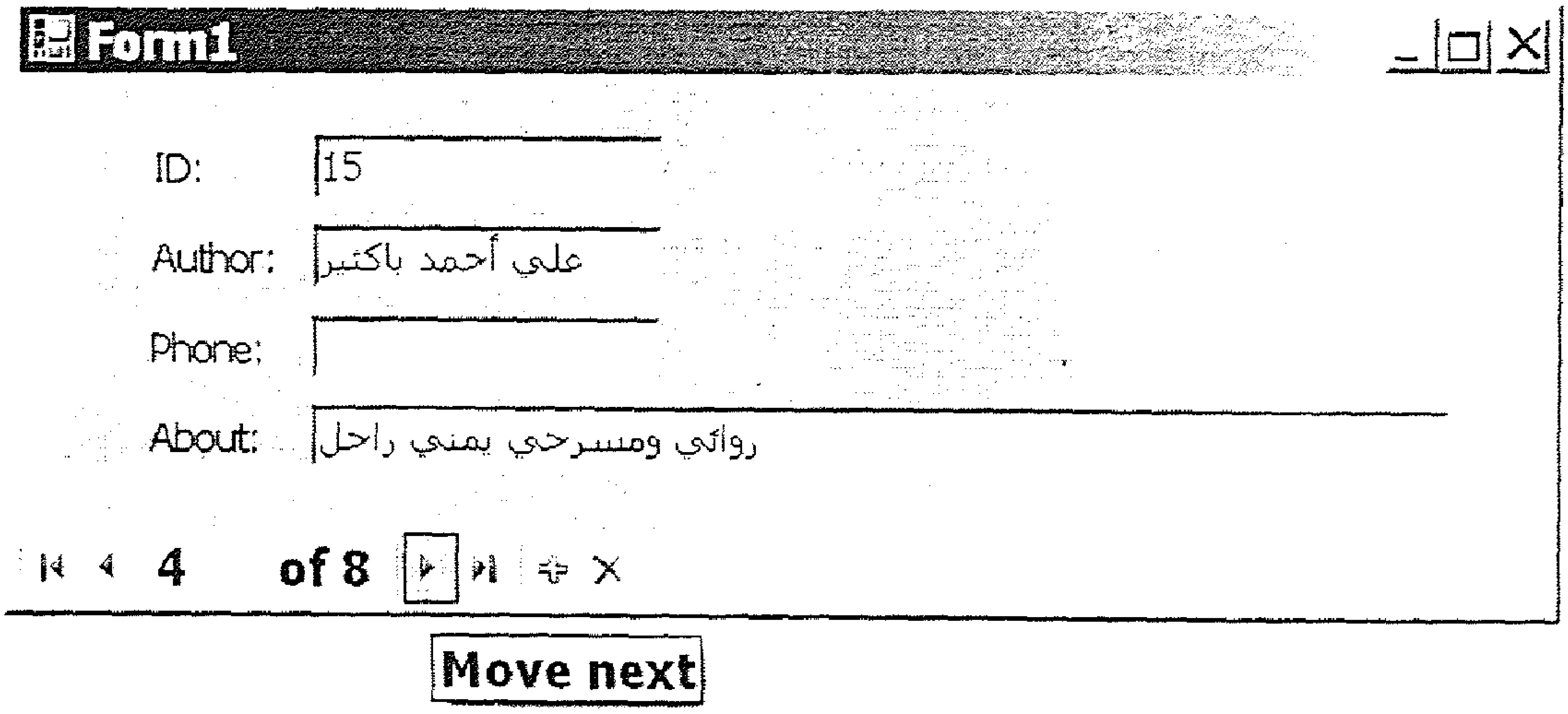
هذه الفئة ترث فئة رف الأدوات ToolStrip Class، لهذا فهـي تعمـل كـرف أدوات يعرض مجموعة من الأزرار، التي تتيح للمستخدم التحرك عبر سـجلات مصدر البيانات وحذف السجل الحالي أو إضافة سجل جديد، كل هـذا بـدون أن تكتب أنت حرفا من الكود!

ويبدو رف الأدوات الذي يعرضه موجه الربط في وقـت التـصميم كمـا فـي الصورة:




لاحظ أن آخر زر على الشريط يتيح لـك إضـافة أزرار وأدوات جديـدة إلـى الشريط، بنفس الطريقة التي تعلمناها في كتاب برمجة الويندوز.. هذا معناه أنـك تستطيع استغلال مساحة الشريط لإضافة أزرار وقـوائم ومربعـات نـصوص ولافتات تؤدي أية وظائف أخرى خاصة بك (كالقص واللصق وعـرض حالـة البرنامج وغير هذا)، وبهذا لا تحتاج إلى إضافة رف أدوات آخر خاص بك.

والصورة التالية تريك كيف يبدو موجه الربط عند تشغيل المشروع Navigator المرفق بأمثلة هذا الكتاب:




مرة أخرى أذكرك: لو أضفت سجلا جديد بضغط زر الإضافة الموجـود علـى شريط موجه الربط، أو حذفت السجل الحالي بضغط زر الحذف، أو غيرت قيمة أي حقل في السجل الحالي بتغيير قيمة أحد مربعات النص، فإن هذه التغييـرات ستؤثر فقط على مجموعة البيانات DataSet، لكن تظل مهمـة تحـديث قاعـدة البيانات متروكة لك.. وإذا كنت لا ترغب أن يعبث المستخدم بقيم بعض الحقول،

فاجعل مربعات النصوص المناظرة لها للقراءة فقط، أو اربط هذه الحقول بلافتات منذ البداية.. مع ملاحظة أن تغيير المستخدم لقيمة المعرف ID لن تؤثر في شيء، لأن هذا الحقل مولد آليا، ومصدر البيانات لا يستطيع تغييره.

ولو لم تكن ترغب في أن يحذف المستخدم السجلات أو يضيف سجلات جديدة، فيمكنك إزالة زر الحذف أو زر الإضافة من فوق الشريط في وقت التصميم، أو يمكنك تعطيلهما، وسترى كيف نفعل هذا بعد قليل ونحن نتعرف على خصائص موجه الربط.

ولحدث إنشاء الفئة BindingNavigator الصيغ التالية:

1- الصيغة الأولى بدون معاملات.
2- الصيغة الثانية تستقبل كائن مصدر الربط BindingSource، الذي سيتحكم من خلاله موجه الربط في سجلات مصدر البيانات.
3- الصيغة الثالثة تستقبل معاملا منطقيا، إذا جعلته False فلن يعرض موجه الربط أزرار التحكم القياسية (أزرار الانتقال وزر الحذف وزر الإضافة).
4- الصيغة الرابعة تستقبل كائنا من نوع الواجهة IContainer (مثل النموذج)، ليتم عرض شريط موجه الربط عليه.

وإضافة إلى ما ترثه من خصائص الفئة ToolStrip، تمتلك الفئة BindingNavigator الخصائص التالية:

### مصدر الربط BindingSource:

تقرأ أو تغير كائن مصدر الربط BindingSource الذي يستخدمه موجه الربط للتحكم في السجلات.

### عنصر إضافة جديد AddNewItem:

تقرأ أو تغير عنصر رف الأدوات ToolStripItem المستخدم لإضافة سجل جديد إلى مصدر البيانات.. وفي الوضع الافتراضي يكون هذا العنصر من النوع ToolStripButton.. ويكون زر الإضافة معطلا على شريط موجه الربط، إذا كانت للخاصية BindingSource.AllowNew القيمة False.
لاحظ أنك قد تجد زر الإضافة معطلا في بعض البرامج.. إذا حدثت معك هذه المشكلة، فيمكنك وضع القيمة Nothing في هذه الخاصية من الكود، أو اختيار القيمة (None) من القائمة المنسدلة في نافذة الخصائص.. هذا سيعطل الوظيفة الآلية للزر، لكنه لن يحذفه من على شريط موجه الربط.. لهذا يمكنك نقره مرتين بالفأرة، وكتابة السطر الوحيد التالي في حدث ضغطه:

**AuthorsBindingNavigator.BindingSource.AddNew( )**

**عنصر الحذف DeleteItem:**

تقرأ أو تغير عنصر رف الأدوات ToolStripItem المستخدم لحذف السجل الحالي من مصدر البيانات.. وفي الوضع الافتراضي يكون هذا العنصر من النوع ToolStripButton.. ويكون زر الحذف معطلا على شريط موجه الربط، إذا كانت للخاصية BindingSource.AllowRemove القيمة False.

وإذا وجدت زر الحذف معطلا في بعض الحالات، فضع القيمة Nothing في هذه الخاصية من الكود، أو (None) من القائمة المنسدلة في نافذة الخصائص، ثم انقر مرتين بالفأرة فوق زر الحذف الموجود على شريط موجه الربط ، واكتب السطر الوحيد التالي في حدث ضغطه:

**AuthorsBindingNavigator.BindingSource.RemoveCurrent( )**

**عنصر التحرك إلى الأول MoveFirstItem:**

تقرأ أو تغير عنصر رف الأدوات ToolStripItem المستخدم للانتقال إلى أول سجل في مصدر البيانات.. وفي الوضع الافتراضي يكون هذا العنصر من النوع ToolStripButton.

**عنصر التحرك إلى الأخير MoveLastItem:**

تقرأ أو تغير عنصر رف الأدوات ToolStripItem المستخدم للانتقال إلى آخر سجل في مصدر البيانات.. وفي الوضع الافتراضي يكون هذا العنصر من النوع ToolStripButton.

**عنصر التحرك إلى التالي MoveNextItem:**

تقرأ أو تغير عنصر رف الأدوات ToolStripItem المستخدم للانتقال إلى السجل التالي في مصدر البيانات.. وفي الوضع الافتراضي يكون هذا العنصر زر رف الأدوات ToolStripButton.

**عنصر التحرك إلى السابق MovePreviousItem:**

تقرأ أو تغير عنصر رف الأدوات ToolStripItem المستخدم للانتقال إلى السجل السابق في مصدر البيانات.. وفي الوضع الافتراضي يكون هذا العنصر من النوع ToolStripButton.

### عنصر الموضع **PositionItem:**

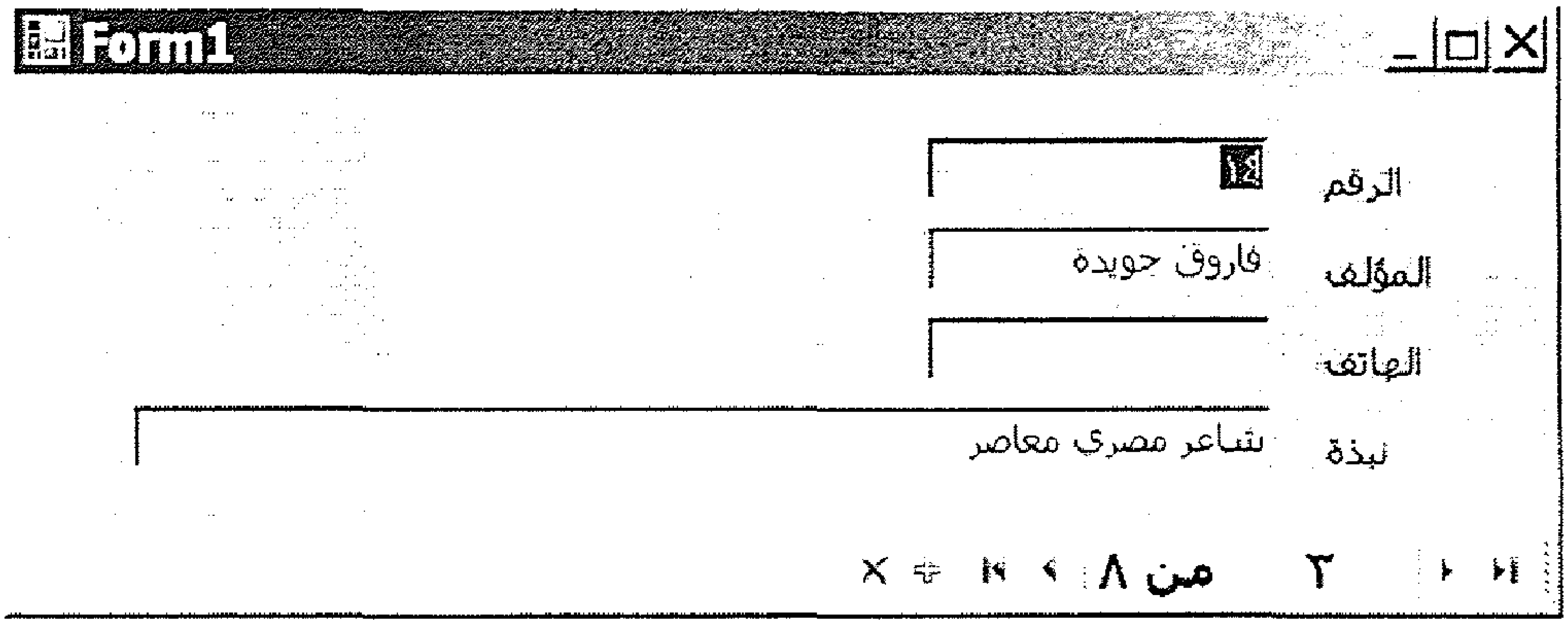
تقرأ أو تغير عنصر رف الأدوات ToolStripItem المستخدم لعرض رقم السجل المعروض حاليا.. وفي الوضع الافتراضي يستخدم مربع نص رف الأدوات ToolStripTextBox لهذا الغرض، وذلك للسماح للمستخدم بكتابة رقم السجل الذي يريده وضغط زر الإدخال Enter للانتقال إليه مباشرة.. ويعرض مربع النص موضع السجل الحالي باستخدام الخاصية BindingSource.Position.

### عنصر العد **CountItem:**

تقرأ أو تغير عنصر رف الأدوات ToolStripItem المستخدم لعرض العدد الكلي للسجلات في مصدر البيانات.. وفي الوضع الافتراضي تستخدم لافتة رف أدوات ToolStripLabel لهذا الغرض، وهي تعرض قيمة الخاصية BindingSource.Count.

### تنسيق عنصر العد **CountItemFormat:**

تستقبل نصا يمثل الصيغة التي سيستخدمها عنصر العد CountItem لعرض عدد السجلات.. وفي الوضع الافتراضي تكون قيمة هذه الخاصية "of {0}".. وفي المشاريع العربية عليك تغييرها إلى صيغة مناسبة، مثل "من {0}" ليبدو الشريط كما في الصورة:




وتمتلك الفئة BindingNavigator الوسائل التالية:

- **إضافة العناصر القياسية AddStandardItems:**
تضيف أزرار الانتقال والحذف والإضافة ومربع نص الموضع ولافتة عدد السجلات إلى شريط موجه الربط.. هذا مفيد إذا أردت إنشاء موجه ربط من الكود وليس في وقت لتصميم.. وفي حالة وجود العناصر القياسية بالفعل على الشريط، فإن هذه الوسيلة لا تحذفها، بل تضيف نسخة أخرى منها، لكنها لا تصير هي العناصر الفعالة.

- **إجازة Validate:**
تجعل النموذج يفحص قيم الأدوات الموجودة عليه، وتعيد True إذا كانت بياناتها صحيحة.

كما تمتلك الفئة BindingNavigator الحدث التالي:

- **إنعاش العناصر RefreshItems:**
ينطلق إذا حدثت تغيرات في مصدر البيانات، تستدعي تحديث أزرار ولافتات موجه الربط.

-١٦-

# جدول عرض البيانات

# DataGridView

توجد الأداة DataGridView تحت الشريط Data في صندوق الأدوات Toolbox، وهي تصلح لعرض جداول قواعد البيانات، وتتيح لك تنسيق البيانات المعروضة بالشكل الذي يناسبك، كما تتيح للمستخدم إضافة السجلات وحذفها، وتغيير قيم خاناتها، حيث تحفظ هذه التعديلات مباشرة في الجدول الأصلي في مجموعة البيانات من خلال آلية الربط، ويمكنك بعد هذا إجراء عملية تحديث Update لإرسال هذه التغييرات إلى قاعدة البيانات.

| **ملحوظة:** |
| --- |
| الأداة DataGridView هي تطوير لأداة قديمة اسمها DataGrid وهي ما زالت متاحة للاستخدام لكنها لا تظهر في شريط الأدوات إلا إذا قمت أنت بإضافتها إليه بالطريقة المألوفة.. وننصح باستخدام الأداة DataGridView لأنها تملك قدرات أكثر بكثير، وإن كانت الأداة DataGrid تتفرد بالقليل من الميزات، لهذا سنتعرف عليها في الفصل التالي. |

ولربط جدول العرض بأحد جداول قاعدة البيانات، يمكنك استخدام الخاصيتين DataSource و DataMember بإحدى الطريقتين التاليتين:

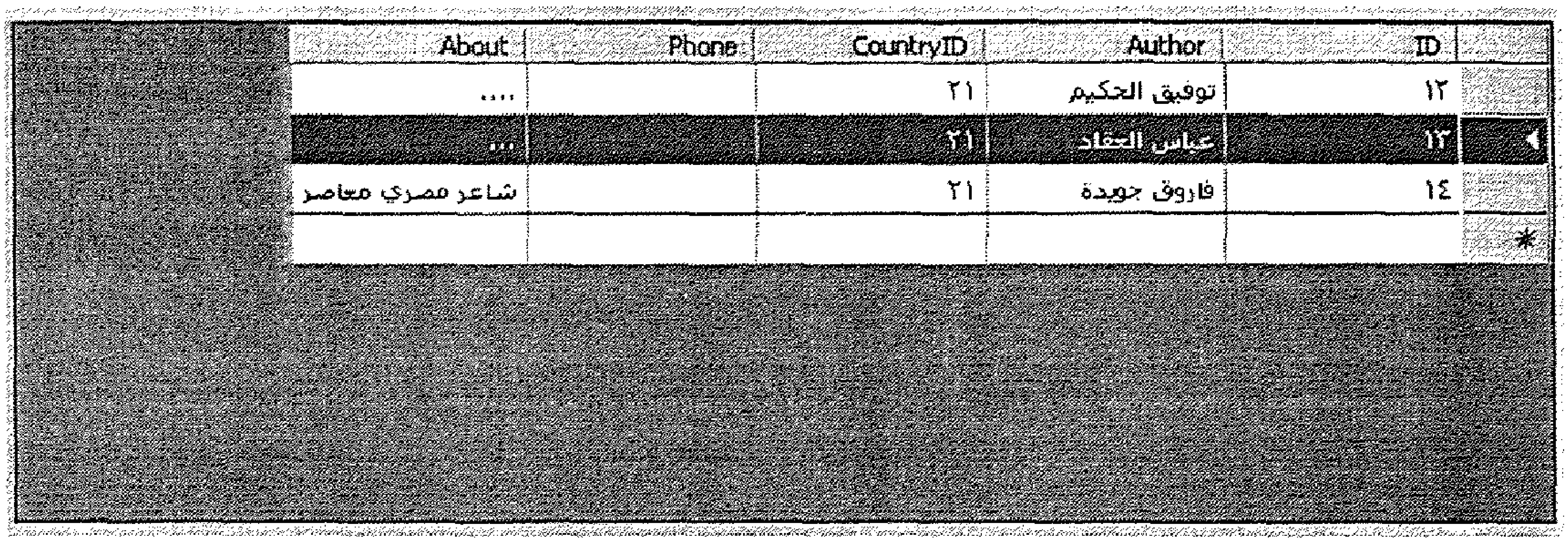
١- أن تضع مجموعة البيانات في الخاصية DataSource، وتضع اسم الجدول في الخاصية DataMember كالتالي:

```
DataGridView1.DataSource = DsBooks
DataGridView1.DataMember = "Authors"
```

٢- أن تضع الجدول مباشرة في الخاصية DataSource كالتالي:

```
DataGridView1.DataSource = DsBooks.Tables("Authors")
```

في كلتا الحالتين سيظهر جدول العرض كما في الصورة:



| About | Phone | CountryID | Author | ID |
|---|---|---|---|---|
| .... | | ٢١ | توفيق الحكيم | ١٢ |
| ... | | ٢١ | عباس العقاد | ١٣ |
| شاعر مصري معاصر | | ٢١ | فاروق جويدة | ١٤ |
| | | | | |

كما تلاحظ في الصورة، يتكون جدول عرض البيانات مما يلي:

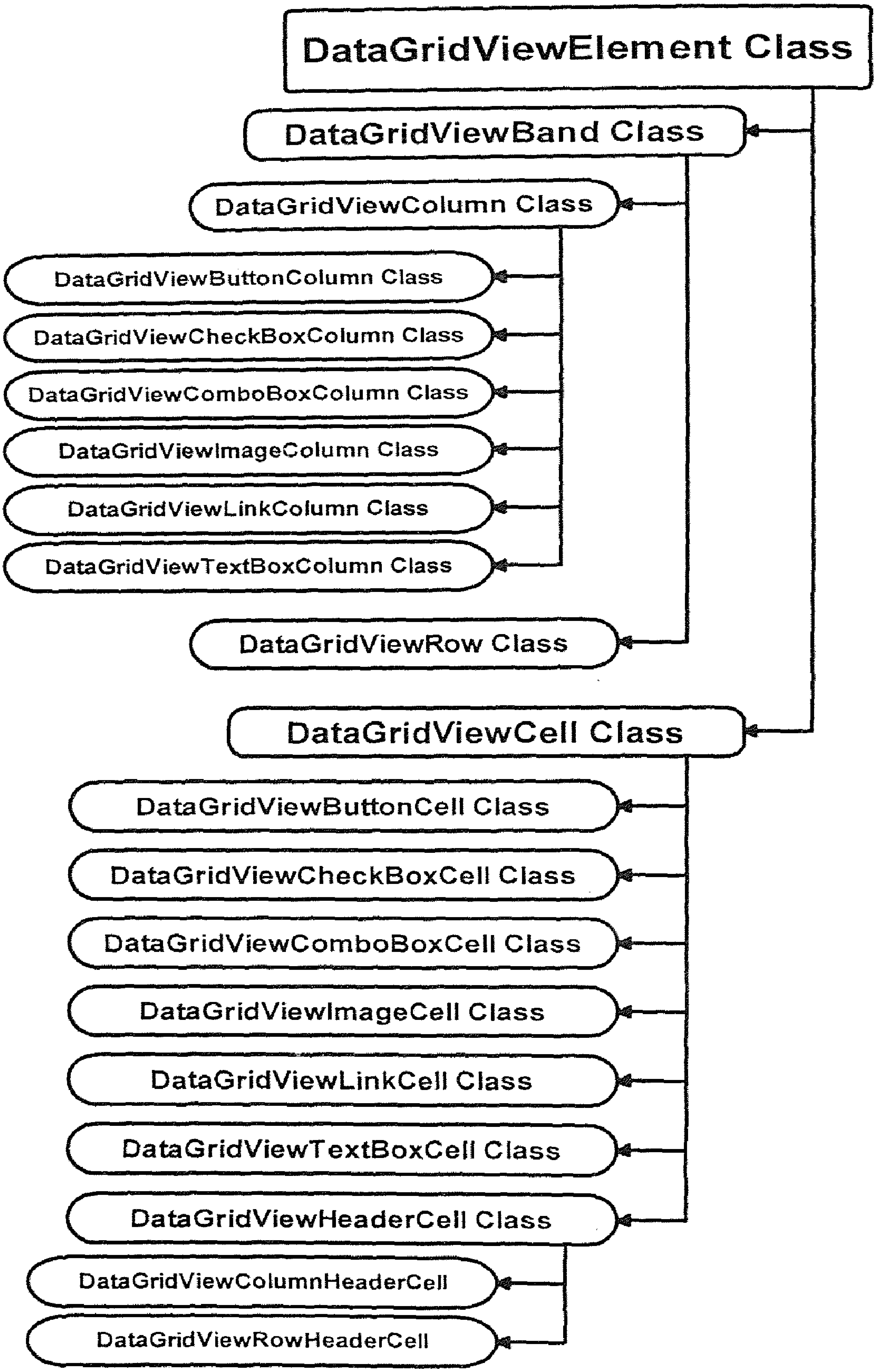
- أعمدة Columns، ولكل عمود منها رأس Header، وهو خانة ثابتة من خانات الجدول تظهر أعلى العمود وتعرض عنوان العمود، لهذا سنسمي رأس العمود أحيانا بخانة العنوان.
- صفوف Rows، ولكل منها هامش عند الضغط عليه يتم تحديد الصف.. هذا الهامش يسمى رأس الصف، أو خانة عنوان الصف، وهو يعرض أيقونة التحرير عند الكتابة في أي خانة في الصف، ويعرض أيقونة الخطأ عند وجود قيم خاطئة في الصف، كما يمكنك أن تكتب فيه نصا كعنوان للصف.. لاحظ أن الصف الأخير (الذي تسبقه النجمة *) هو صف جديد New Row، عندما يكتب فيه المستخدم تتم إضافته إلى الجدول، ويظهر صف جديد بدلا منه.
- خلايا Cells: وهي خانات الجدول، ويمكنك ضغطها بالفأرة لبدء تحريرها، كما يتم بدء التحرير بمجرد الكتابة من لوحة المفاتيح أثناء تحديد الخانة.. وعندما تكون الخانة في وضع التحرير، يظهر فيها مربع نص للكتابة فيه.. ويمكن إنهاء التحرير بضغط Enter أو إلغائه بضغط Esc لتعود القيمة الأصلية للخانة.. ولا يتم قبول التغييرات التي حدثت في خانات أحد الصفوف إلا إذا ضغط المستخدم CTRL+Enter، أو انتقل إلى صف آخر.
- خلفية جدول العرض، وهي المنطقة الخالية التي لا تظهر فيها الخانات.

ويمكنك ربط جدول العرض بمصادر بيانات أخرى غير جداول البيانات.. والمشروع BindGridToArray يريك كيف يمكن ربط جدول العرض بمصفوفة تحتوي على كائنات من نوع الفئة Student، وبسطر واحد من الكود:

**Grd.DataSource = Std**

كل ما فعلناه هو استخدام المصفوفة كمصدر للبيانات، ليقوم جدول العرض تلقائيا بإنشاء أعمدة بأسماء خصائص الفئة Student، ووضع قيم المصفوفة فيها.. منتهى البساطة والروعة!

لكن لا تدع هذه البساطة تخدعك، فجدول العرض أداة ضخمة، وهناك عدد كبير من الفئات Class لتمثيل مكوناتها، كما هو موضح في مخطط الوراثة التالي:




دعنا إذن نتعرف على هذه الفئات أولا، ثم نتعرف على الفئة DataGridView.

## فئة عنصر جدول العرض DataGridViewElement Class

هذه هي الفئة الأم التي تشتق منها كل عناصر جدول العرض: الأعمدة والصفوف والخانات، وهي تملك خاصيتين فقط:

### جدول العرض DataGridView:

تعيد كائن جدول العرض DataGridView الذي ينتمي إليه العنصر.

### الحالة State:

تعيد إحدى قيم المرقم DataGridViewElementStates التي توضح حالة العنصر، من بين القيم التالية:

| | |
|---|---|
| None | العنصر في حالته الافتراضية. |
| Frozen | العنصر مجمد (مثبت).. هذا معناه أنه يظل ظاهرا في موضعه مهما تحرك المستخدم بالمنزلق. |
| ReadOnly | العنصر للقراءة فقط، ولا يستطيع المستخدم تغيير قيمته. |
| Resizable | يمكن للمستخدم تغيير موضع وحجم العنصر في الجدول.. لاحظ أن هذه القيمة يتم تجاهلها إذا لم يتم دمجها مع القيمة ResizableSet. |
| ResizableSet | العنصر مستقل في قابلية تغيير حجمه، عن العنصر الرئيسي الذي ينتمي إليه. |
| Selected | العنصر محدد حاليا Highlighted. |
| Displayed | العنصر ظاهر للمستخدم حاليا دون الحاجة إلى تحريك المنزلق لعرضه. |
| Visible | العنصر مرئي (غير مختفي).. هذا صحيح حتى لو كان العنصر غير معروض Displayed على الشاشة بسبب موضعه من المنزلق. |

ويعمل هذا المرقم كمؤشر Flag، لهذا يمكن أن تعيد هذه الخاصية أكثر من قيمة مدمجة معا، وعليك فحص القيمة التي تريدها باستخدام المعامل And:

```
If (DataGridView1.Rows(0).State And
        DataGridViewElementStates.Displayed) > 0 Then
   MsgBox(DataGridView1.Rows(0).State.ToString)
End If
```

## فئة نطاق جدول العرض DataGridViewBand Class

هذه الفئة ترث الفئة DataGridViewElement، كما أنها تمثل الواجهتين ICloneable و IDisposable.
وتعمل هذه الفئة كفئة أم تشتق منها أعمدة وصفوف جدول العرض، ولعل هذا يوضح سبب تسمية هذه الفئة باسم النطاق Band، فهي تمثل نطاقا من الخانات تقع في صف معين أو عمود معين.

وتمتلك هذه الفئة الخصائص التالية:

### رف القائمة الموضعية ContextMenuStrip:

تقرأ أو تغير كائن رف القائمة الموضعية ContextMenuStrip الذي يعرض القائمة الموضعية للنطاق الحالي.. هذا معناه أنك تستطيع استخدام قائمة موضعية مختلفة لكل صف، ولكل عمود!.. وتظهر القائمة الموضعية كما تعرف عند الضغط بزر الفأرة الأيمن فوق أي خانة في النطاق، ما عدا خانة راس النطاق.
لاحظ أن أولوية القائمة الموضعية تكون كالتالي:
١- تظهر القائمة الموضعية للخانة إن وجدت.
٢- تظهر القائمة الموضعية للصف إن وجدت.
٣- تظهر القائمة الموضعية للعمود إن وجدت.
هذا معناه أن الخانة التي توجد في صف له قائمة موضعية وعمود له قائمة موضعية، ستعرض القائمة الموضعية الخاصة بالصف وليس العمود.

### النوع الافتراضي لخانة العنوان DefaultHeaderCellType:

ضع في هذه الخاصية كائن النوع Type الذي يمثل نوع خانة العنوان (رأس العمود)، بشرط أن تكون قيمة هذه الخاصية من نوع الفئة DataGridViewHeaderCell أو أي فئة مشتقة منها.. وسنتعرف على هذه الفئات بالتفصيل لاحقا.
وتفيدك هذه الخاصية إذا أردت تغيير شكل ووظيفة خانة رأس الصف أو العمود.. يمكنك مثلا تعريف فئة جديدة مشتقة من الفئة DataGridViewColumnHeaderCell، ومنحها الشكل والأداء الذي تريده، ثم وضع نوعها في الخاصية DefaultHeaderCellType لكل عمود في جدول العرض، لتظهر كخانة رأس لكل منها.

### مجمد Frozen:

إذا جعلت قيمة هذه الخاصية True، فسيتم تثبيت النطاق الحالي في موضعه مهما حرك المستخدم المنزلق الأفقي أو الرأسي.. هذا مفيد إذا أرت استخدام أحد الصفوف أو الأعمدة كعنوان ثابت بحيث يظل مرئيا باستمرار.
لاحظ أنك تستطيع تثبيت النطاق (الصف أو العمود) إذا كان أول نطاق أو يسبقه نطاق مثبت، وغير هذا يحدث خطأ في البرنامج.

### رقم العنصر Index:

تعيد رقم النطاق الحالي في الجدول.. فإذا كان النطاق الحالي صفا، تعيد موضعه في مجموعة صفوف الجدول، وإن كان النطاق الحالي عمودا، تعيد موضعه الأصلي في مجموعة الأعمدة.
لاحظ أن تغيير المستخدم لموضع العمود بسحبه بالفأرة (إذا كنت تسمح له بترتيب الأعمدة) لا يؤثر على رقم العمود في مجموعة الأعمدة، وإنما يؤثر فقط على موضع عرضه DisplayIndex.. لكن على العكس، يتغير موضع الصف في مجموعة الصفوف، إذا ضغط المستخدم رأس أحد الأعمدة لترتيب الصفوف على أساسه.. لهذا لا تحتفظ برقم الصف في متغير طوال تشغيل البرنامج، لأنه قد يتغير في أي لحظة، وبدلا من هذا احتفظ بمتغير من النوع DataGridViewRow يشير إلى الصف الذي تريده.

### الطراز الافتراضي للخانة DefaultCellStyle:

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في شكل خانات النطاق الحالي.. وسنتعرف على الفئة DataGridViewCellStyle بالتفصيل لاحقا.

### يوجد طراز افتراضي للخانة HasDefaultCellStyle:

تعيد True إذا كنت قد وضعت قيمة في الخاصية DefaultCellStyle..
لاحظ أن الخاصية DefaultCellStyle لا تعيد Nothing أبدا، فلو كانت فارغة وحاولت قراءتها، فسيتم إنشاء طراز افتراضي ووضعه فيها!.. لهذا يمكنك استخدام الخاصية HasDefaultCellStyle أولا لمعرفة إن كانت الخاصية DefaultCellStyle فارغة أم لا.

### الطراز الموروث InheritedStyle:

تعيد كائن طراز الخانة DataGridViewCellStyle الذي يتم تطبيقه على النطاق الحالي.. وتعيد هذه الخاصية الطراز الموضح في الخاصية DefaultCellStyle إذا كانت لها قيمة، وإلا فإنها تعيد الطراز الموروث من جدول العرض.

### للقراءة فقط ReadOnly:

إذا جعلت قيمة هذه الخاصية True، فلن يستطيع المستخدم تغيير قيمة أي خانة في النطاق الحالي.

### قابل للتحجيم Resizable:

تحدد إن كان بإمكان المستخدم تغيير حجم النطاق الحالي (تغيير عرض العمود أو ارتفاع الصف) باستخدام الفأرة.. وهي تأخذ إحدى قيم المرقم DataGridViewTriState التالية:

| | |
|---|---|
| NotSet | لم يتم تحديد قيمة الخاصية، وسيتم استخدام القيمة الافتراضية الموروثة من جدول العرض. |
| True | توضع القيمة True في الخاصية. |
| False | توضع القيمة False في الخاصية. |

### محدد Selected:

إذا جعلت قيمة هذه الخاصية True، فسيتم تحديد النطاق الحالي.. كما تعيد هذه الخاصية True إذا كان النطاق الحالي محددا سواء بواسطتك أو بواسطة المستخدم.. لاحظ أن تحديد النطاق الحالي لا يلغي تحديد النطاقات الأخرى، فجدول العرض يتيح تحديد أكثر من صف أو عمود معا.

لاحظ أن هذه الخاصية تتأثر بقيمة الخاصية SelectionMode الخاصة بجدول العرض كالتالي:

- إذا كانت للخاصية SelectionMode أي قيمة غير FullRowSelect و RowHeaderSelect، فإن الخاصية Select الخاصة بالصف تكون بلا تأثير لأن جدول العرض لا يسمح بتحديد الصفوف.
- إذا كانت للخاصية SelectionMode أي قيمة غير FullColumnSelect و ColumnHeaderSelect، فإن الخاصية Select الخاصة بالعمود تكون بلا تأثير لأن جدول العرض لا يسمح بتحديد الأعمدة.

**معروض Displayed:**

تعيد True إذا كان النطاق الحالي ظاهرا للمستخدم على الشاشة دون الحاجة إلى استخدام المنزلق الأفقي أو الرأسي.

**مرئي Visible:**

إذا جعلت قيمة هذه الخاصية False، فسيتم إخفاء النطاق الحالي وعدم عرضه في الجدول.

**الوسم Tag:**

هذه هي الخاصية الإضافية، التي تستطيع أن تضع فيها أي كائن يحوي معلومات تهمك تتعلق بالنطاق الحالي.

## فئة أساس المجموعة BaseCollection Class

هذه الفئة تمثل الواجهة ICollection، وهي تعمل كمجموعة عادية، ولكنها لا تحتوي على أية وسائل تتيح لك إضافة العناصر إليها.. كل ما تحتويه هو العناصر التالية، وهي مألوفة لنا فقد شرحناها بالتفصيل في كتاب برمجة إطار العمل، لهذا لن نعيد شرحها هنا:

| | |
|---|---|
| **Count** | **IsReadOnly** |
| **IsSynchronized** | **SyncRoot** |
| **CopyTo** | **GetEnumerator** |

## فئة مجموعة أعمدة جدول العرض
## DataGridViewColumnCollection Class

هذه الفئة ترث الفئة BaseCollection كما أنها تمثل واجهة القائمة IList، وهي تحتوي على عناصر من نوع فئة عمود جدول العرض DataGridViewColumn Class.

ولحدث إنشاء هذه المجموعة صيغة واحدة، تستقبل كائن جدول العرض DataGridView الذي ستنتمي إليه مجموعة الأعمدة.. مثال:

**Dim Cols As New DataGridViewColumnCollection(
DataGridView1)**

في الحقيقة، لا يبدو إرسال جدول العرض كمعامل إلى حدث ذا مغزى، فبعد تنفيذ الجملة السابقة لن يتغير شيء في مجموعة الأعمدة الخاصة بجدول العرض DataGridView1، ولن تكون له أية صلة بالمجموعة الجديدة Cols، والتي بدورها ستكون فارغة ولن تحتوي على أية أعمدة موجودة حاليا في جدول العرض DataGridView1!!

وتمتلك مجموعة الأعمدة العناصر التقليدية للمجموعات، والتي تستقبل كمعامل كائن العمود أو نصا يمثل اسم العمود.. لهذا دعنا نركز هنا على العناصر التالية:

### إضافة Add:

تضيف عمودا إلى مجموعة الأعمدة، وتعيد رقما صحيحا يمثل موضع هذا العمود في المجموعة.. ولهذه الوسيلة الصيغتان التاليتان:

١- الصيغة الأولى تستقبل كائن العمود DataGridViewColumn وتضيفه إلى المجموعة.

٢- الصيغة الثانية تستقبل نصا يمثل اسم العمود، ونصا يمثل عنوان العمود، وتنشئ عمودا نصيا DataGridViewTextBoxColumn وتضيفه إلى المجموعة.. والمثال التالي يضيف إلى جدول العرض عمودا نصيا اسمه Col1 وعنوانه "عمود١":

**DataGridView1.Columns.Add("Col1", "عمود١")**

وتتسبب هذه الوسيلة في حدوث خطأ في البرنامج في الحالات التالية:

- إذا كان العمود المراد إضافته موجودا في جدول العرض من قبل.
- إذا كان الجدول يحتوي على صف أو أكثر، بينما للخاصية CellType الخاصة بالعمود القيمة Nothing.

- إذا كان العمود الجديد مثبتا Frozen، وأضفته وسط أعمدة غير مثبتة.
- إذا كان جدول العرض يقوم بتحديد كل خاناته في تلك اللحظة أو يزيل تحديدها أو يغير قيم الخاصية DisplayIndex لكل الأعمدة.
- إذا تم استدعاء الوسيلة Add من داخل أي من الأحداث التالية:
CellEnter, CellLeave, CellValidating, CellValidated, RowEnter, RowLeave, RowValidated, RowValidating.
- إذا كانت للخاصية SortMode الخاصة بالعمود القيمة Automatic، بينما للخاصية DataGridView.SelectionMode القيمة FullColumnSelect أو ColumnHeaderSelect.
- إذا كانت للخاصية InheritedAutoSizeMode الخاصة بالعمود القيمة ColumnHeader، بينما عناوين الأعمدة غير معروضة (DataGridView.ColumnHeadersVisible = False).
- إذا كانت للخاصية InheritedAutoSizeMode القيمة Fill، بينما العمود مثبتا (Frozen = True).

### العنصر Item:

هذه هي الخاصية الافتراضية، وهي تعيد كائن العمود DataGridViewColumn الذي ترسل إليها اسمه أو رقمه كمعامل..
والكود التالي يعرض رقم العمود Col1 الذي أضفناه في المثال السابق:

**MsgBox(DataGridView1.Columns("Col1").Index)**

### معرفة عدد الأعمدة GetColumnCount:

تعيد عدد أعمدة الجدول التي لها الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates.. والمثال التالي يعرض عدد الأعمدة المحددة في جدول العرض:

```
MsgBox(DataGridView1.Columns.GetColumnCount(
        DataGridViewElementStates.Selected))
```

### معرفة عرض الأعمدة GetColumnsWidth:

تعيد مجموع عروض أعمدة الجدول التي لها الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates.. والمثال التالي يخبرك بعرض الأعمدة المرئية في الجدول:

```
MsgBox(DataGridView1.Columns.GetColumnsWidth(
        DataGridViewElementStates.Visible))
```

### معرفة أول عمود GetFirstColumn:

تستقبل إحدى قيم المرقم DataGridViewElementStates وتعيد أول عمود له الحالة المرسلة.. وتعيد هذه الوسيلة Nothing إذا لم تجد عمودا له الحالة المطلوبة.

وتوجد صيغة أخرى لهذه الوسيلة لها معامل ثانٍ هو أيضا من نوع المرقم DataGridViewElementStates، ولكنه يستقبل الحالة التي يجب ألا يكون عليها العمود.. والمثال التالي يعيد أول عمود مرئي لكنه غير معروض للمستخدم:

```
Dim DgCol As DataGridViewColumn =
    DataGridView1.Columns.GetFirstColumn(
    DataGridViewElementStates.Visible,
    DataGridViewElementStates.Displayed)
```

### معرفة آخر عمود GetLastColumn:

مماثلة للوسيلة السابقة، ولكنها تعيد آخر عمود له الحالة الموضحة في المعامل الأول وليست له الحالة الموضحة في المعامل الثاني.

### معرفة العمود التالي GetNextColumn:

تعيد كائن العمود DataGridViewColumn الذي يحقق الشروط الموضحة في المعاملات، وهي بالترتيب:

١- كائن العمود DataGridViewColumn الذي سيبدأ البحث منه، للعثور على أول عمود يليه يحقق الشروط المطلوبة.

٢- إحدى قيم المرقم DataGridViewElementStates توضح الحالة التي يجب أن يمتلكها العمود المطلوب.

٣- إحدى قيم المرقم DataGridViewElementStates توضح الحالة التي يجب ألا يمتلكها العمود المطلوب.

وتعيد هذه الوسيلة Nothing إذا لم تجد عمودا يحقق الشروط المطلوبة.

والمثال التالي يعرض أسماء كل الأعمدة الظاهرة للمستخدم والتي لا يستطيع تغيير حجمها:

```
Dim Cols = DataGridView1.Columns
Dim DgCol = Cols.GetFirstColumn(
    DataGridViewElementStates.Displayed,
    DataGridViewElementStates.Resizable)
```

```
Do Until DgCol Is Nothing
    MsgBox(DgCol.Name)
    DgCol = Cols.GetNextColumn(DgCol,
              DataGridViewElementStates. Displayed,
              DataGridViewElementStates.Resizable)
Loop
```

**معرفة العمود السابق GetPreviousColumn:**

مماثلة للوسيلة السابقة في معاملاتها، ولكنها تبحث عن العمود السابق للعمود المرسل للمعامل الأول، الذي يحقق الشروط المطلوبة.. دعنا نعيد كتابة المثال السابق باستخدام هذه الوسيلة مع الوسيلة GetLastColumn، لعرض أسماء الأعمدة بترتيب عكسي:

```
Dim Cols = DataGridView1.Columns
Dim DgCol = Cols.GetLastColumn(
     DataGridViewElementStates.Displayed,
     DataGridViewElementStates.Resizable)
Do Until DgCol Is Nothing
    MsgBox(DgCol.Name)
    DgCol = Cols.GetPreviousColumn(DgCol,
              DataGridViewElementStates. Displayed,
              DataGridViewElementStates.Resizable)
Loop
```

**المجموعة تغيرت CollectionChanged:**

ينطلق هذا الحدث عند حدوث تغير في عناصر المجموعة بالحذف أو الإضافة.. والمعامل الثاني e لهذا الحدث من النوع CollectionChangeEventArgs وقد تعرفنا عليه من قبل عند التعرف على مجموعة الجداول DataTableCollection.

## فئة عمود جدول العرض DataGridViewColumn Class

هذه الفئة ترث الفئة DataGridViewBand وتمثل الواجهة IComponent.
وتعمل هذه الفئة كعمود في جدول عرض البيانات، ولحدث إنشائها صيغتان:

١- الصيغة الأولى بدون معاملات.
٢- والصيغة الثانية تستقبل كائن خانة DataGridViewCell لوضعه في الخاصية CellTemplate الخاصة بالعمود.. هذه الخانة ستعمل كقالب Template تنسخ منه كل خانة تضاف إلى العمود عند إضافة صف جديد إلى الجدول.

وتمتلك هذه الفئة الخصائص التالية:

### الاسم Name:

تقرأ أو تغير اسم العمود.. هذا الاسم لا يظهر كعنوان له، وإنما يستخدم كمعرف للعمود داخل مجموعة الأعمدة، لاستخدامه مع بعض الوسائل مثل Remove و Contains.

### نوع القيمة ValueType:

ضع في هذه الخاصية كائن النوع Type، الذي يمثل نوع بيانات خانات العمود الحالي.

### نص تلميح الأداة ToolTipText:

ضع في هذه الخاصية النص الذي تريد عرضه للمستخدم عندما يحلق بالفأرة للحظات فوق رأس العمود.

### العرض Width:

تقرأ أو تغير عرض العمود الحالي، والقيمة الافتراضية لها هي ١٠٠.

### أقل عرض MinimumWidth:

تقرأ أو تغير أقل عرض ممكن للعمود، بحيث لا يمكن تصغيره عنه برمجيا أو عند سحب المستخدم لحافته بالفأرة لتغيير حجمه.. والقيمة الافتراضية لهذه الخاصية هي ٥، وغير مسموح لك بتصغيرها عن ٢.

### طريقة الحجم التلقائي AutoSizeMode:

تحدد كيف يتم تغيير حجم العمود تلقائيا، وهي تأخذ إحدى قيم المرقم DataGridViewAutoSizeColumnMode التالية:

| | |
|---|---|
| NotSet | طريقة تحجيم العمود موروثة من جدول العرض. |
| None | لا يتم ضبط عرض العمود تلقائيا. |
| AllCells | ضبط عرض العمود ليلائم محتويات كل خاناته، بما في ذلك خانة العنوان. |
| AllCells ExceptHeader | ضبط عرض العمود ليلائم محتويات كل خاناته، ما عدا خانة العنوان. |
| DisplayedCells | ضبط عرض العمود ليلائم محتويات كل خاناته الظاهرة على الشاشة حاليا، بما في ذلك خانة العنوان. |
| DisplayedCells ExceptHeader | ضبط عرض العمود ليلائم محتويات كل خاناته الظاهرة على الشاشة حاليا، ما عدا خانة العنوان. |
| ColumnHeader | ضبط عرض العمود ليلائم محتويات خانة العنوان.. وتسبب هذه القيمة خطأ في البرنامج إذا كان جدول العرض يخفي عناوين الأعمدة (ColumnHeadersVisible = False). |
| Fill | ضبط عرض العمود الحالي مع باقي الأعمدة لمحاولة ملء مساحة جدول العرض كلها.. وتسبب هذه القيمة خطأ في البرنامج إذا كان العمود مثبتا Frozen. |

لاحظ أن المستخدم يستطيع جعل العمود يأخذ الحجم المناسب لمحتوياته، بمجرد النقر مرتين بالفأرة فوق الحافة اليمنى لخانة العنوان.

### طريقة الحجم التلقائي الموروثة InheritedAutoSizeMode:

مماثلة للخاصية السابقة مع وجود اختلاف واحد، فلو كانت للخاصية AutoSizeMode القيمة NotSet، فإن الخاصية InheritedAutoSizeMode تعيد القيمة الموروثة من جدول العرض.

### أولوية الملء FillWeight:

تقرأ أو تغير الوزن النسبي لكل عمود، لاستخدامه في معرفة كيفية ملء مساحة جدول العرض، وذلك عندما تكون للخاصية InheritedAutoSizeMode القيمة Fill.. والقيمة الافتراضية لهذه الخاصية هي ١٠٠، ويمكنك زيادتها أو إنقاصها، حيث يتم تكبير العمود الذي له وزن أكبر، أكثر من العمود الذي وزن أصغر.. الذي يحدث هو حساب مجموع قيم هذه الخاصية لكل الأعمدة (وليكن Sum)، ثم ضرب كل عمود في القيمة (FillWeight/Sum).. لاحظ أن أقصى قيمة للمجموع Sum يجب ألا تزيد عن ٦٥٥٣٥ وإلا حدث خطأ في البرنامج، لهذا لا تضع قيما كبيرة في هذه الخاصية، فالعبرة ليست في كبر القيمة، ولكن العبرة في كبر النسبة إلى المجموع.. فلتكن جميع القيم أصغر من ١٠٠.

### طريقة الترتيب SortMode:

تقرأ أو تغير طريقة ترتيب خانات الجدول، وهي تأخذ إحدى قيم المرقم DataGridViewColumnSortMode التالية:

| | |
|---|---|
| Automatic | هذه هي القيمة الافتراضية للجداول التي تعرض خانات نصية، وهي تسمح للمستخدم بضغط رأس العمود بالفأرة، لترتيب صفوف الجدول تبعا خانات هذا العمود.. ويظهر في خانة العنوان مثلث يشير رأسه إلى اتجاه الترتيب، ويمكن تغيير اتجاه الترتيب بضغط رأس العمود بالفأرة مرة أخرى.. وتسبب هذه القيمة خطأ في البرنامج إذا كانت للخاصية SelectionMode الخاصة بدول العرض القيمة FullColumnSelect أو ColumnHeaderSelect. |
| NotSortable | لا يمكن للمستخدم إجراء عملية الترتيب، ورغم أنه ما يزال بإمكانك إجراء الترتيب برمجيا، فلن يحتوي رأس العمود على مساحة لعرض علامة الترتيب.. وهذه هي القيمة الافتراضية للجداول التي تعرض خانات تحتوي على أزرار أو صور أو مربعات اختيار أو قوائم منسدلة أو وصلات. |
| Programmatic | مماثلة للقيمة السابقة، لكن رأس العمود سيحتوي على مساحة لعرض علامة الترتيب. |

وتستطيع منح كل عمود في الجدول طريقة ترتيب مختلفة عن غيره.. يمكنك مثلا أن تسمح للمستخدم بترتيب الجدول عندما يضغط رأس العمود الذي يعرض المفتاح الأساسي فقط، بينما تجعل باقي الأعمدة غير قابلة للترتيب.

### هل هو مرتبط بالبيانات IsDataBound:

تعيد True إذا كان العمود الحالي مرتبطا بمصدر بيانات.

### اسم خاصية البيانات DataPropertyName:

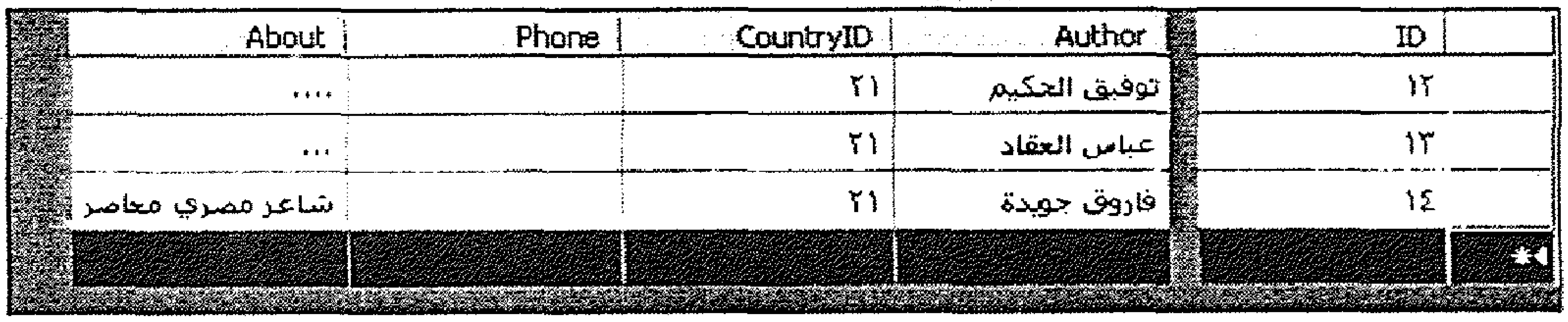
تحدد اسم العمود الأصلي في مجموعة البيانات، الذي يحفظ فيه العمود الحالي بياناته.. وتأخذ هذه الخاصية قيمته تلقائيا عند ربط جدول العرض بجدول في مجموعة البيانات، وإن كان باستطاعتك تغيير قيمة هذه الخاصية يدويا لربطها بأي عمود تريده، أو أي عنصر بيانات.

### رقم العرض DisplayIndex:

تقرأ أو تغير الموضع الذي يظهر فيه العمود الحالي في جدول العرض.. هذا لا يؤثر في شيء على ترتيب العمود في مجموعة الأعمدة، والذي توضحه الخاصية Index.

### عرض الفاصل DividerWidth:

تقرأ أو تغير حجم الخط الذي يفصل العمود الحالي عن العمود التالي.. هذا الفاصل هو مساحة خالية تأخذ نفس لون أرضية جدول العرض، ورغم أنها تعتبر جزءا من العمود الحالي، إلا أنها لا تؤدي أية وظيفة من وظائفه، ما عدا العمل كفاصل شكلي.. والصورة التالية توضح تأثير وضع القيمة ١٠ في هذه الخاصية في العمود الأول:



| ID | Author | CountryID | Phone | About |
| --- | --- | --- | --- | --- |
| ١٢ | توفيق الحكيم | ٢١ | | .... |
| ١٣ | عباس العقاد | ٢١ | | ... |
| ١٤ | فاروق جويدة | ٢١ | | شاعر مصري معاصر |

والقيمة الافتراضية لهذه الخاصية هي صفر، لهذا لا يوجد أي فاصل بين الأعمدة، ما عدا خطوط الشبكة العادية.

### خانة العنوان HeaderCell:

ضع في هذه الخاصية كائن خانة عنوان العمود DataGridViewColumnHeaderCell الذي يحمل خصائص رأس العمود الحالي.. وسنتعرف على هذه الفئة بالتفصيل لاحقا.

**نص العنوان HeaderText:**

تقرأ أو تغير النص الذي يظهر في خانة عنوان العمود الحالي.

**قالب الخانة CellTemplate:**

ضع في هذه الخاصية كائن الخانة DataGridViewCell الذي تريد استخدامه كقالب تستمد منه الخانات التي تضاف إلى العمود خصائصها.

**نوع الخانة CellType:**

تعيد كائن النوع Type، الذي يوضح نوع الخانة المستخدمة كقالب في الخاصية CellTemplate.. لاحظ أن الخاصية CellTemplate من النوع الفئة الأم DataGridViewCell، بينما تعيد الخاصية CellType النوع الفعلي المشتق من هذه الفئة الأم.. وسنتعرف على الفئة DataGridViewCell ومشتقاتها لاحقا.

كما تمتلك هذه الفئة الوسيلة التالية:

**معرفة العرض المفضل GetPreferredWidth:**

تعيد أنسب عرض للعمود تبعا للمواصفات المطلوبة، وهي تأخذ معاملين:

- إحدى قيم المرقم DataGridViewAutoSizeColumnMode، تحدد طريقة الحجم التلقائي للعمود.
- معامل منطقي، إذا جعلت قيمته True، فسيتم تقدير العرض المناسب للعمود بافتراض أن ارتفاع خاناته سيظل ثابتا، أما إذا جعلته False، فسيدخل في الاعتبار إمكانية تغيير ارتفاعات الصفوف، وهل هناك التفاف لأسطر الخانات Word wrap أم لا، حيث ستتم المحافظة على النسبة بين عرض العمود وارتفاع خاناته.

والفئة DataGridViewColumn تعمل كفئة أم لكل من الفئات التالية:

١- عمود مربعات النصوص DataGridViewTextBoxColumn.

٢- عمود الأزرار DataGridViewButtonColumn.

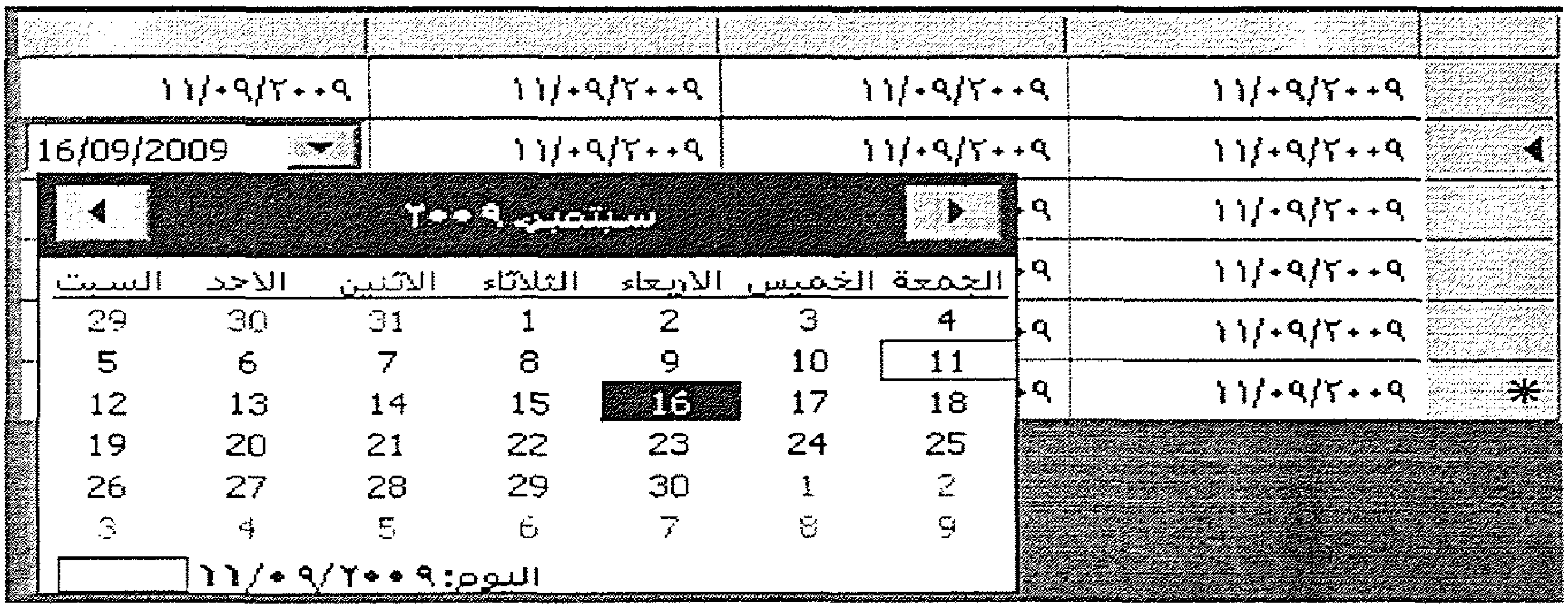
٣- عمود مربعات الاختيار DataGridViewCheckBoxColumn.

٤- عمود القوائم المركبة DataGridViewComboBoxColumn.

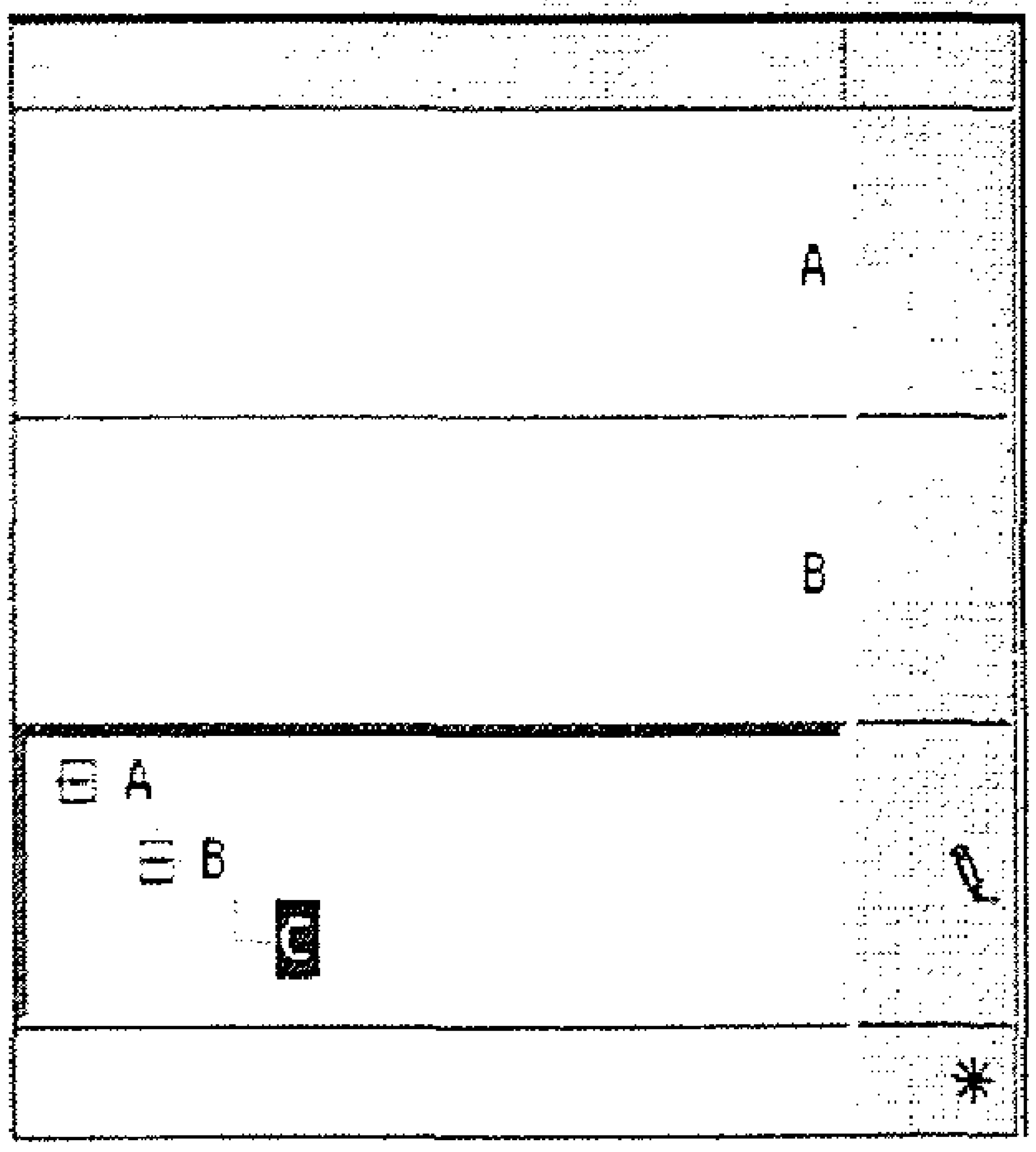
٥- عمود الصور DataGridViewImageColumn.

٦- عمود الوصلات DataGridViewLinkColumn.

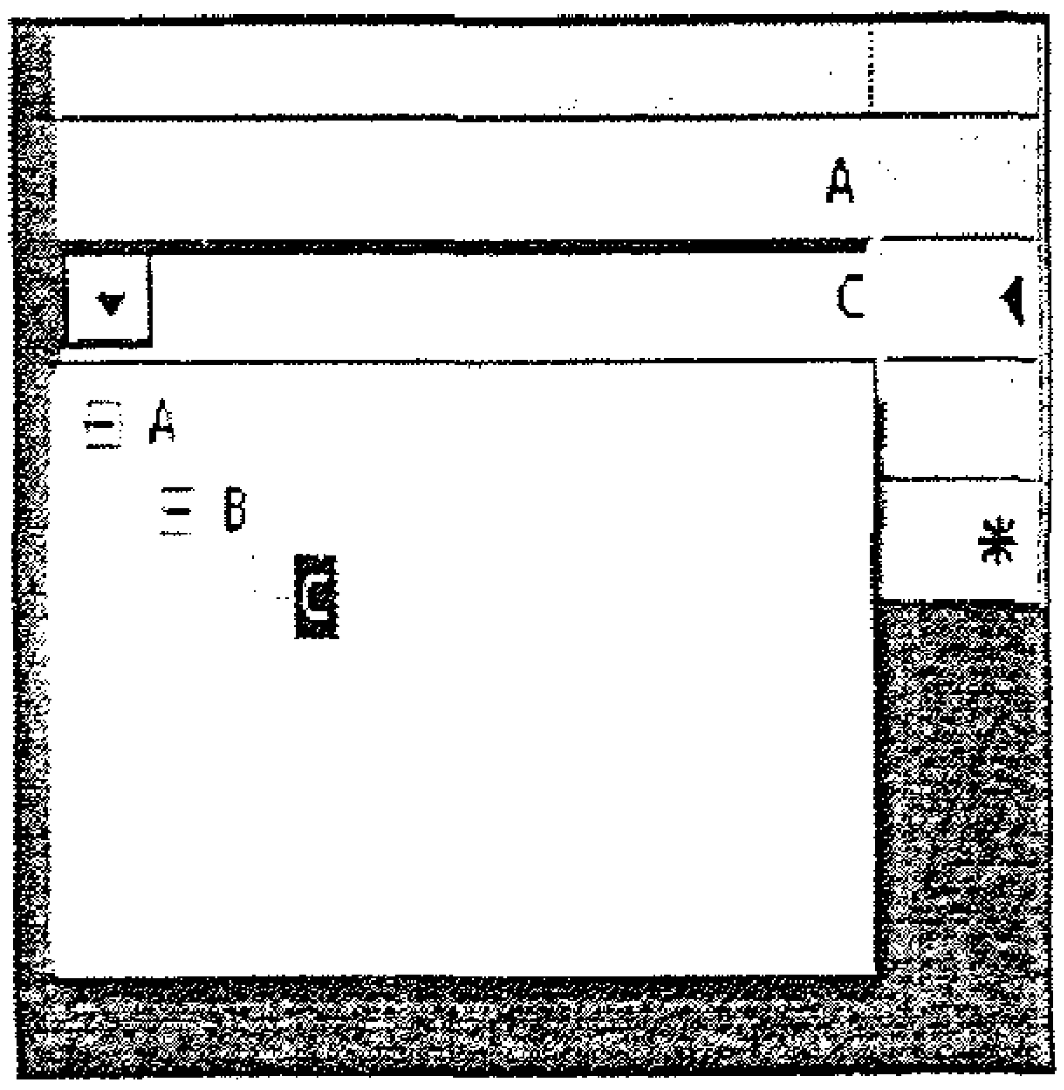
ولكن الأمر لا يتوقف عند هذه الأنواع، فبإمكانك وراثة هذه الفئة لإنشاء أعمدة تعرض خاناتها أي أداة أخرى من أدوات الويندوز.. ولو ضغطت الزر "عمود تواريخ" في المشروع DataGridColumnTypes فسيضاف إلى جدول العرض عمود تعرض كل خانة من خاناته أداة اختيار التاريخ DateTimePicker.




ولو فتحت متصفح المشاريع، لوجدت فيه فئة اسمها CalendarColumn ترث الفئة DataGridViewColumn.. وكود هذه الفئة بسيط للغاية، فهي تستبدل Override عنصرين فقط من عناصر الفئة الأم: حدث الإنشاء New والخاصية CellTemplate، وذلك للتعامل مع النوع الجديد لخانات هذا العمود، وهو فئة جديدة أنشأناها بأنفسنا أيضا اسمها CalenderCell مهمتها عرض أداة اختيار التاريخ.. وسنتعرف على فكرة هذه الفئة لاحقا.



وبنفس الطريقة، أمكننا إنشاء عمود تعرض كل خانة فيه شجرة TreeView، ويمكنك إضافة هذا العمود إلى الجدول بضغط الزر "عمود أشجار" في نفس المشروع.



لاحظ أن هذه الشجرة غير عملية، فهي تظهر بكاملها داخل الخانة، وهو ما يحتاج إلى جعل مساحة الخانة كبيرة لضمان ظهور فروع الشجرة بشكل مقبول.. ويمكنك حل هذه المشكلة باستخدام شجرة منسدلة، وقد شرحنا فكرتها في كتاب "برمجة نماذج الويندوز" في المشروع TreeComboBox.. على كل حال، يمكنك

إضافة هذا النوع من الأعمدة إلى جدول العرض بضغط الـــزر "عمـــود أشـــجار منسدلة".

## عمود مربعات النصوص
## DataGridViewTextBoxColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناته من النوع DataGridViewTextBoxCell، وهي خانات تعرض لافتات، وعند تحرير أي خانة منها، فإنها تعرض مربع نص.. ويمكنك ضغط الزر "عمود النصوص" في المشروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.

ويعتبر عمود مربعات النص النوع الافتراضي الذي يضيفه جدول العرض عند ربطه بمصدر بيانات.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخاصية التالية:

### أقصى طول للمدخلات MaxInputLength:

تحدد أقصى عدد من الحروف تقبله كل خانة في العمود.. والقيمة الافتراضية لهذه الخاصية هي ٣٢٧٦٧، ولو جعلتها صفرا فهذا يعني السماح للمستخدم بكتابة الحد الأقصى من الحروف، وهو يتجاوز ٢ مليار حرف.

## فئة عمود الأزرار
## DataGridViewButtonColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناته من النوع DataGridViewButtonCell، وهي خانات تحمل كل منها زرا Button يمكن للمستخدم ضغطه.. ويمكنك ضغط الزر "عمود أزرار" في المشروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

### النص Text:
تقرأ أو تغير النص الافتراضي المعروض على جميع الأزرار الموجودة في خانات العمود.

### استخدام نص العمود لقيمة الزر UseColumnTextForButtonValue:
إذا جعلت قيمة هذه الخاصية False (وهي القيمة الافتراضية)، فلن يظهر النص الموجود في الخاصية Text على أزرار خانات العمود، وستكون كل خانة مسئولة عن وضع النص الخاص بالزر الذي تعرضه.

### طريقة العرض المسطح FlatStyle:
تحدد طريقة عرض الزر، وهي تأخذ إحدى قيم المرقم FlatStyle التالية:

| | |
|---|---|
| Standard | يظهر الزر مجسما بالطريقة القياسية المعتادة ثلاثية الأبعاد.. هذه هي القيمة الافتراضية. |
| Flat | يظهر الزر مسطحا، ويتغير لون خلفيته عند مرور الفأرة فوقه وعند ضغطه. |
| Popup | يظهر الزر مسطحا، لكن عند المرور فوقه بالفأرة يبرز إلى أعلى ويصير مجسما. |
| System | يظهر الزر تبعا لاختيارات المستخدم ضمن نظام الويندوز على جهازه. |

# فئة عمود مربعات الاختيار

# DataGridViewCheckBoxColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناته من النوع DataGridViewCheckBoxCell، وهي خانات تحمل مربع اختيار CheckBox.. ويمكنك ضغط الزر "عمود مربعات اختيار" في المشروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.
ولحدث إنشاء هذه الفئة صيغتان:

١- الصيغة الأولى بدون معاملات.
٢- والصيغة الثانية تستقبل معاملا منطقيا Boolean، يتم إرسال قيمته إلى الخاصية ThreeState التي سنتعرف عليها بعد قليل.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

### طراز العرض المسطح FlatStyle:

مماثلة لتلك الخاصة بعمود الأزرار.

### ثلاثي الحالة ThreeState:

إذا جعلت قيمة هذه الخاصية True فسيكون مربع الاختيار ثلاثي الحالة (متضمنا الحالة الوسيطة غير المحددة Indeterminate).. هذا معناه أن المستخدم إذا ضغط المربع مرة فستوضع به علامة الاختيار، وإذا ضغطه مرة أخرى فسيصير في الحالة الوسيطة (بعلامة اختيار غائمة) وإذا ضغطه مرة ثالثة فستزال علامة الاختيار.
أما إذا جعلت قيمتة هذه الخاصية False (وهذه هي القيمة الافتراضية)، فسيكون لمربع الاختيار حالتان فقط (Checked – Unchecked)، وهذا معناه أن المستخدم إذا ضغط مربع الاختيار مرة فستوضع به علامة الاختيار، وإذا ضغطه مرة أخرى فستزال منه هذه العلامة.

### القيمة الخاطئة FalseValue:

تستقبل كائنا Object يحتوي على القيمة المناظرة لحالة عدم الاختيار Unchecked.

### القيمة الصحيحة TrueValue:

تستقبل كائنا Object يحتوي على القيمة المناظرة لحالة الاختيار Checked.

### القيمة غير المحددة IndeterminateValue:

تستقبل كائنا Object يحتوي على القيمة المناظرة للحالة الوسيطة غير المحددة Indeterminate.

ويمكنك استخدام الخصائص FalseValue و TrueValue و IndeterminateValue إذا كان العمود الحالي مرتبطا بمصدر بيانات.. افرض على سبيل المثال أن لديك مصدر بيانات يحتوي على ثلاثة أرقام هي ٠، ١، ٢، وقمت بربطها بعمود مربعات اختيار.. في هذه الحالة افعل ما يلي:

- ضع في الخاصية FalseValue القيمة ٠ لتخبر العمود أن الخانات المناظرة للقيمة ٠ في مصدر البيانات لن توضع بها علامات الاختيار.
- ضع في الخاصية TrueValue القيمة ١ لتخبر العمود أن الخانات المناظرة للقيمة ١ في مصدر البيانات ستوضع بها علامات الاختيار.
- ضع في الخاصية IndeterminateValue القيمة ٢ لتخبر العمود أن الخانات المناظرة للقيمة ٢ في مصدر البيانات ستوضع بها علامات اختيار غائمة دلالة على أنها حالة وسيطة غير محددة.

# فئة عمود الصور
# DataGridViewImageColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناته من النوع DataGridViewImageCell، وهي خانات تعرض كل منها صورة.. ويمكنك ضغط الزر "عمود الصور" في المشروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.
ولحدث إنشاء هذه الفئة صيغتان:

١- الصيغة الأولى بدون معاملات.
٢- والصيغة الثانية تستقبل معاملا منطقيا Boolean، يتم إرسال قيمته إلى الخاصية valuesAreIcons التي سنتعرف عليها بعد قليل.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

### الوصف Description:
ضع في هذه الخاصية نصا يصف الصور أو الأيقونات الموجودة في خانات العمود.

### القيم أيقونات ValuesAreIcons:
إذا جعلت قيمة هذه الخاصية True، فستعرض خانات العمود الأيقونة الموجودة في الخاصية Icon.. أما إذا جعلتها False (وهي القيمة الافتراضية)، فستعرض خانات العمود الصورة الموجودة في الخاصية Image.

### الأيقونة Icon:
ضع في هذه الخاصية كائن الأيقونة Icon الذي تريد عرضه في خانات العمود عندما تكون للخاصية ValuesAreIcons القيمة True.

### الصورة Image:
ضع في هذه الخاصية كائن الصورة Image الذي تريد عرضه في خانات العمود عندما تكون للخاصية ValuesAreIcons القيمة False.

### مخطط الصورة ImageLayout:

تحدد طريقة عرض الصورة في خانات العمود، وهي تأخذ إحدى قيم المرقم DataGridViewImageCellLayout التالية:

| | |
|---|---|
| NotSet | القيمة غير محددة (متروكة لكل خانة على حدة). |
| Normal | يتم عرض الصورة كاملة في منتصف كل خانة.. هذه هي القيمة الافتراضية. |
| Stretch | يتم مط الصورة لتلائم عرض وارتفاع كل خانة.. هذا معناه أن الصورة ستملأ كل مساحة الخانة، ولكن هذا قد يؤدي إلى تشويهها. |
| Zoom | يتم تكبير أو تصغير الصورة لتلائم عرض أو ارتفاع الخانة، مع المحافظة على النسبة الأصلية بين ارتفاع الصورة وعرضها، مما يمنع تشويهها. |

لاحظ أن جدول البيانات في الوضع الافتراضي يعرض عمود طابع الوقت TimeStamp تلقائيا في عمود صور، لمجرد أن هذا العمود يحمل بيانات ثنائية.. وقد رأينا كيف سبب لنا هذا مشاكل كثيرة فيما سبق.

ولحل هذه المشكلة، عليك تغير نوع هذا العمود بعد ربط جدول العرض بمجموعة البيانات.. لفعل هذا في وقت التصميم، اتبع الخطوات التالية:

- حدد جدول العرض على النموذج وافتح نافذة الخصائص.
- اضغط الرابط Edit Columns الموجود في الجزء السفلي من نافذة الخصائص.. سيفتح هذا نافذة تحرير مجموعة الأعمدة.
- حدد عمود طابع الوقت، لعرض خصائصه في الجزء الأيمن من النافذة.
- حدد الخاصية ColumnType تحت الشريط Design، واضغط زر الإسدال، واختر من القائمة النوع DataGridTextBoxColumn بدلا من النوع DataGridImageColumn.. هذا سيمنع الأخطاء التي تحدث بسبب محاولة رسم البيانات الثنائية كصورة.. لاحظ أن العمود لا يملك فعليا خاصية اسمها ColumnType، وما يفعله المصمم هو حذف العمود القديم، وإنشاء عمود جديد من النوع الذي اخترته في هذه الخاصية مع نسخ باقي خصائص العمود القديم إلى العمود الجديد.. لهذا لا تستطيع تغيير نوع العمود من الكود، إلا بحذفه وإنشاء عمود جديد.
- أو يمكنك أن تضع القيمة False في الخاصية Visible لإخفاء العمود وحل المشكلة من جذورها، وهذا ما فعلناه من الكود في المشاريع السابقة:

**DGAuthors.Columns("RowVersion").Visible = False**

## فئة عمود الوصلات DataGridViewLinkColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناته من النوع DataGridViewLinkCell، وهي خانات تعرض كل منها رابطا (وصلة) Link.. ويمكنك ضغط الزر "عمود الوصلات" في المشروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

**النص Text:**

تقرأ أو تغير نص الوصلة المعروضة في خانات العمود.

**استخدم نص العمود كقيمة للوصلة UseColumnTextForLinkValue:**

إذا جعلت قيمة هذه الخاصية False (وهي القيمة الافتراضية)، فلن يعرض النص الموجود في الخاصية Text كوصلات في خانات العمود، وستكون كل خانة مسئولة عن كتابة نص الوصلة التي تعرضها.

**سلوك الرابط LinkBehavior:**

تحدد كيف يبدو شكل الرابط، وهي تأخذ إحدى قيم المرقم LinkBehavior:

| | |
|---|---|
| AlwaysUnderline | الرابط تحته خط دائما. |
| HoverUnderline | لا يوضع خط تحت الرابط إلا حينما يمر فوقه مؤشر الفأرة. |
| NeverUnderline | لا يوضح خط تحت الرابط مطلقا. |
| SystemDefault | خيارات متصفح الإنترنت ونظام الويندوز هي التي توضح كيف يبدو الرابط. |

**لون الرابط LinkColor:**

تحدد لون الروابط المعروضة في العمود.. وفي الوضع التلقائي يكون هذا اللون هو الأزرق.

**لون الرابط الفعال ActiveLinkColor:**
تحدد لون الرابط أثناء ضغطه بالفأرة.. في الوضع التلقائي يكون هذا اللون هو الأحمر.

**لون الرابط المُزار VisitedLinkColor:**
تحدد لون الرابط بعد أن يضغطه المستخدم.. في الوضع التلقائي يكون هـذا اللون هو الأحمر الغامق.

**تتبع حالة الزيارة TrackVisitedState:**
إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتغير لون الرابط بعد ضغطه، ليأخذ اللون المحدد في الخاصية VisitedLinkColor.

وقد استخدمنا عمود الوصلات في المشروع CustomDataSet لعرض وصـلة "عرض درجات الطالب".

## فئة عمود القوائم المركبة
## DataGridViewComboBoxColumn Class

هذه الفئة ترث الفئة DataGridViewColumn، وهي تعمل كعمود خاناته من النوع DataGridViewComboBoxCell، وهي خانات تحمل قوائم منسدلة Combo Boxes.. ويمكنك ضغط الزر "عمود قوائم مركبة" في المشروع DataGridColumnTypes لإضافة عمود من هذا النوع إلى جدول العرض.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

### طراز العرض المسطح FlatStyle:

مماثلة لتلك الخاصية بعمود الأزرار.

### عرض القائمة المنسدلة DropDownWidth:

تحدد عرض القائمة المسندلة.. والقيمة الافتراضية لهذه الخاصية هي ١ ولا يقبل أقل منها، لكن عليك أن تلاحظ أن وضع أي قيمة أصغر من عرض العمود في هذه الخاصية سيجعلها بدون تأثير، لأن القائمة المنسدلة يمكن فقط أن تكون مساوية لعرض العمود أو أكبر منه!.. هذا يضمن لك أن القائمة المنسدلة ستأخذ نفس عرض العمود تلقائيا لو قام المستخدم بزيادة عرضه.. هذا معناه أن عرض القائمة المنسدلة يحسب فعليا من العلاقة التالية:

Actual Width = Max (DropDownWidth, Col Width)

### إكمال تلقائي AutoComplete:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتم اقتراح التكملة المناسبة للحروف التي يكتبها المستخدم في القائمة المركبة.

### مصدر البيانات DataSource:

ضع في هذه الخاصية الكائن الذي سيعمل كمصدر بيانات، لاستخدامه في ملء القائمة المركبة بالعناصر، مثل اسم جدول الدول Countries:

```
Col.DataSource = Ds.Tables("Countries")
```

لاحظ أن العمود في هذه الحالة يتعامل مع مصدرين من مصادر البيانات:

- مصدر بيانات جدول العرض نفسه (الذي يرتبط بجدول المؤلفين مثلا).
- مصدر بيانات القائمة المركبة (الذي يرتبط بجدول الدول).

### عنصر العرض DisplayMember:

ضع في هذه الخاصية اسم عنصر البيانات، لعـرض قيمتـه فـي القائمـة المركبة.. مثل العمود Name في جدول الدول:

```
Col.DisplayMember = "Name"
```

| ملحوظة: |
| --- |
| لو استخدمت مصفوفة نصية كمصدر للبيانات، فلست بحاجة إلى الخاصيتين DisplayMember فعناصر المصفوفة نفسها ستكون عناصر العـرض، وهذا هو ما فعلناه عند وضع القيم في عمود القوائم المركبة في المـشروع DataGridColumnTypes.. لاحظ وأنت تجرب المشروع أن ضـغط زر إسدال القائمة لا يعمل إلا إذا كانت الخانة التي يوجد فيها محددة أولا، لهذا فإن أول ضغطة ستعمل على تحديد الخانة، وثاني ضـغطة ستـسدل القائمة. |

### عنصر القيمة ValueMember:

تعمل هذه الخاصية مع الخاصية DataPropertyName الموروثة من فئة العمود، للربط بين جدولين.. مثلا: في المشروع UpdateErrors2 حـذفنا العمود CountryID وأضفنا بدلا منه عمود قوائم مركبة يعـرض أسـماء الدول، بدلا من أن نعرض للمستخدم رقم الدولة التي ينتمي إليها المؤلف:

```
' حذف عمود أرقام الدول
DataGridView1.Columns.Remove("CountryID")
' تعريف عمود قوائم منسدلة
Dim Col As New DataGridViewComboBoxColumn()
Col.Name = "Country"
' ملء العمود بأسماء الدول
Col.DataSource = Ds.Tables("Countries")
Col.DisplayMember = "Name"
' إضافة العمود على جدول العرض
DataGridView1.Columns.Insert(2, Col)
```

هذا سهل ومفهوم، فالقائمة الآن مرتبطة بجدول الدول وتعرض قيم الحقـل Name.. لكن المشكلة أن المستخدم لو اختار الدولة مـصر مـثلا، فعلينـا وضع رقم هذه الدولة (وهو ١٢) فـي الحقـل CountryID فـي جـدول المؤلفين، وهذا قد يحتاج إلى كتابة بعض الكود يدويا.

لا تقلق.. لن نكتب أي كود لفعل هذا.. كل ما علينا هو إخبار القائمة المركبة أنها ستحفظ قيمة العمود ID التابع لمصدر بيانات العمود الحالي (وهو هنا جدول الدول)، في العمود CountryID التابع لمصدر البيانات جدول العرض كله (وهو هنا جدول المؤلفين)، وذلك كالتالي:

**Col.ValueMember = "ID"**
**Col.DataPropertyName = "CountryID"**

**العناصر Items:**

تعيد مجموعة من النوع ObjectCollection، وهي مجموعة تمثل الواجهة IList معرفة داخل الفئة DataGridViewComboBoxCell، وكل عنصر من عناصرها من النوع Object، وهي تحتوي على عناصر القائمة المركبة.. لاحظ إن استخدام الخاصية DataSource يلغي عمل الخاصية Items.. مثلا: لو أضفت بعض العناصر إلى المجموعة Items ثم وضعت مصدر بيانات في الخاصية DataSource، فستعرض القائمة المنسدلة عناصر مصدر البيانات وتتجاهل العناصر التي أضفتها إلى الخاصية Items.. أما إذا وضعت مصدر البيانات في الخاصية DataSource أولا ثم حاولت إضافة بعض العناصر إلى الخاصية Items فسيحدث خطأ في البرنامج!

**طريقة العرض DisplayStyle:**

تحدد كيف تظهر القائمة المركبة في خانات العمود، وهي تأخذ إحدى قيم المرقم DataGridViewComboBoxDisplayStyle التالية:

| | |
|---|---|
| ComboBox | حتى عندما لا تكون الخانة محددة، ستظل تعرض القائمة المركبة. |
| DropDownButton | عندما لا تكون الخانة محددة، ستعرض زر إسدال القائمة المركبة بمفرده. |
| Nothing | عندما لا تكون الخانة محددة، فإنها لا تعرض القائمة المركبة ولا حتى زر إسدالها، ولا تظهر القائمة المركبة إلا في الخانة المحددة فقط. |

**طريقة عرض الخانة الحالية فقط DisplayStyleForCurrentCellOnly:**

إذا جعلت قيمة هذه الخاصية True، فستؤثر الخاصية DisplayStyle على الخانة المحددة في العمود الحالي فقط دون باقي خانات العمود.. والقيمة الافتراضية لهذه الخاصية هي False.

لاحظ أن وضع القيمة Nothing في الخاصية DisplayStyle مع وضع القيمة True في الخاصية DisplayStyleForCurrentCellOnly سيخفي القائمة المركبة وزر الإسدال من كل خانات العمود بما في ذلك الخانة المحددة!.. في هذه الحالة على المستخدم نقر الخانة المحددة مرتين بالفأرة لإظهار القائمة المركبة!

**أقصى عدد من العناصر المسدلة MaxDropDownItems:**

تقرأ أو تغير عدد العناصر التي يمكن عرضها في القائمة المنسدلة بدون الحاجة إلى منزلق رأسي، أما إذا زاد عدد العناصر عن هذا العدد فسيظهر منزلق رأسي ليتيح للمستخدم عرض باقي العناصر.. وتقبل هذه الخاصية قيمة ما بين ١ و ١٠٠، وقيمتها الافتراضية ٨.

**مرتبة Sorted:**

إذا جعلت قيمة هذه الخاصية True، فسيتم ترتيب عناصر القائمة المرتبة أبجديا.

| **ملحوظة:** |
|---|
| الخصائص السابقة تؤثر على جميع القوائم المركبة الموجودة في كل خانات العمود، كما تؤثر على القائمة المركبة الخاصة بقالب الخانة الموجود في الخاصية CellTemplate.. لكن يظل بإمكانك تغيير خصائص القائمة المركبة لكل خانة على حدة، كما سنرى لاحقا. |

## فئة مجموعة صفوف جدول العرض
## DataGridViewRowCollection Class

هذه المجموعة تمثل الواجهة IList، وهي تحتوي على عناصر من نوع صف جدول العرض DataGridViewRow.
ويستقبل حدث إنشاء هذه الفئة كائن جدول العرض DataGridView الذي تنتمي إليه المجموعة.. لكنك لا تحتاج إلى إنشاء نسخة جديدة منها، لأنك تتعامل مع مجموعة صفوف جدول العرض من خلال الخاصية Rows الخاصة به.

وإضافة إلى العناصر التقليدية للمجموعات، تمتلك هذه الفئة الوسائل التالية:

### إضافة Add:
تضيف صفا جديدا إلى جدول العرض وتعيد عددا صحيحا يمثل موضعه في مجموعة الصفوف، ولها الصيغ التالية:
١- الصيغة الأولى بدون معاملات، وهي تضيف صفا له القيم الافتراضية.
٢- الصيغة الثانية تستقبل كائن الصف DataGridViewRow لإضافته إلى المجموعة.
٣- الصيغة الثالثة تستقبل مصفوفة كائنات Object Array، وتنشئ صفا جديدا وتضع في خاناته قيم هذه المصفوفة بنفس الترتيب، وتضيفه إلى جدول العرض.

وتتسبب هذه الوسيلة في حدوث خطأ في البرنامج في الحالات التالية:
- إذا لم تكن هناك أية أعمدة في جدول العرض.
- إذا كان عدد خانات الصف أكبر من عدد أعمدة الجدول.
- إذا كان الصف الجديد مثبتا Frozen، لكنه سيضاف بعد صفوف غير مثبتة.
- إذا كان جدول العرض يقوم بتحديد كل خاناته في تلك اللحظة أو يزيل تحديدها.
- إذا تم استدعاء الوسيلة Add من داخل أي من الأحداث التالية:
CellEnter, CellLeave, CellValidating, CellValidated, RowEnter, RowLeave, RowValidated, RowValidating.
- إذا كان جدول العرض في الوضع الافتراضي (DataGridView.VirtualMode = True).

- إذا كان جدول العرض مرتبطا بمصدر بيانات (DataGridView.DataSource <> Nothing).

لاحظ أن الصف الجديد الذي أضفته لا يتم ترتيبه ضمن صفوف الجدول، ولو أردت أن يوضع في الترتيب الصحيح، فعليك استدعاء الوسيلة DataGridView.Sort في الحدث DataGridView.RowsAdded لإعادة ترتيب صفوف الجدول.

- **إضافة نسخة AddCopy:**

تستقبل رقم صف في مجموعة الصفوف، حيث تنسخ صفا جديدا مماثلا له وتضيفه إلى المجموعة، وتعيد موضع إضافته.

- **إضافة نسخ AddCopies:**

مماثلة للوسيلة السابقة، إلا أنها تنسخ من الصف الموضح رقمه في المعامل الأول، عدد النسخ الموضح في المعامل الثاني، وتضيف هذه الصفوف المنسوخة إلى المجموعة، وتعيد رقم آخر صف تمت إضافته.

- **إدراج Insert:**

تدرج صفا أو أكثر في موضع معين داخل مجموعة الصفوف.. ولهذه الوسيلة عدة صيغة، كلها تشترك في أن معاملها الأول يستقبل الموضع الذي سيتم إدراج الصف فيه، وتختلف في المعامل الثاني كما يلي:

١- يستقبل المعامل الثاني للصيغة الأولى عدد الصفوف الجديدة التي سيتم إدراجها في المجموعة.

٢- يستقبل المعامل الثاني للصيغة الثانية مصفوفة كائنات Object Array، حيث يتم إنشاء صف جديد وإضافته إلى موضع الإدراج، ووضع قيم المصفوفة في خانات هذا الصف بنفس الترتيب.

٣- يستقبل المعامل الثاني للصيغة الثالثة كائن الصف DataGridViewRow المراد إدراجه.

- **إدراج نسخة InsertCopy:**

تنسخ صفا من المجموعة وتدرجه في موضع معين فيها، ولها معاملان:

- المعامل الأول يستقبل رقم الموضع المراد إدراج الصف الجديد فيه.
- المعامل الثاني يستقبل رقم الصف المراد نسخه.

**إدراج نسخ InsertCopies:**
مشابهة للوسيلة السابقة، إلا أنها تزيد عليها بمعامل ثالث، يستقبل عدد النسخ التي سيتم إنشاؤها من الصف الموجود في الموضع المحدد في المعامل الثاني، ليتم إدراجها في مجموعة الصفوف بدءا من الموضع المحدد في المعامل الأول.

**معرفة حالة الصف GetRowState:**
تخبرك بحالة الصف الذي أرسلت إليها رقمه في المجموعة كمعامل.. وهي تعيد إحدى قيم المرقم DataGridViewElementStates الذي تعرفنا عليه من قبل عند التعرف على الخاصية DataGridViewElement.State.

**معرفة أول صف GetFirstRow:**
تعيد موضع أول صف له الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates.. وتعيد هذه الوسيلة -١ إذا لم تجد صفا له الحالة المطلوبة.
وتوجد صيغة أخرى لهذه الوسيلة لها معامل ثانٍ هو أيضا من نوع المرقم DataGridViewElementStates، ولكنه يستقل الحالة التي يجب ألا يكون عليها الصف.. والمثال التالي يعيد رقم أول صف معروض في جدول العرض لكنه غير محدد:

```
MsgBox(DataGridView1.Rows.GetFirstRow(
        DataGridViewElementStates.Displayed,
        DataGridViewElementStates.Selected))
```

**معرفة آخر صف GetLastRow:**
مماثلة للصيغة الأولى للوسيلة السابقة، ولكنها تعيد موضع آخر صف له الحالة المرسلة كمعامل.

**معرفة الصف التالي GetNextRow:**
تعيد رقم الصف الذي له حالة معينة، وهي تستقبل المعاملين التاليين:
١- رقم الصف الذي سيبدأ البحث منه.
٢- إحدى قيم المرقم DataGridViewElementStates توضح الحالة التي يجب أن يمتلكها الصف المطلوب.

وتوجد صيغة ثانية لهذه الوسيلة، لها معامل ثالث، يستقبل قيمة من قيم المرقم DataGridViewElementStates توضح الحالة التي يجب ألا يمتلكها الصف المطلوب.

والمثال التالي يعرض أرقام كل الصفوف المعروضة والتي لا يستطيع المستخدم تغيير حجمها:

```
Dim Rows = DataGridView1.Rows
Dim Pos = Rows.GetFirstRow(
          DataGridViewElementStates.Displayed,
          DataGridViewElementStates.Resizable)
Do Until Pos = -1
   MsgBox(Pos)
   Pos = Rows.GetNextRow(Pos,
         DataGridViewElementStates.Displayed,
         DataGridViewElementStates.Resizable)
Loop
```

- **معرفة الصف السابق GetPreviousRow:**

مماثلة للوسيلة السابقة في صيغتيها، ولكنها تبحث في المجموعة من الخلف إلى الأمام، بدءا من الموضع الذي أرسلته إلى المعامل الأول.. دعنا نكتب المثال السابق باستخدام هذه الوسيلة، لعرض أرقام الصفوف بترتيب عكسي:

```
Dim Rows = DataGridView1.Rows
Dim Pos = Rows.Count -1
Do
   Pos = Rows.GetPreviousRow(Pos,
         DataGridViewElementStates.Displayed,
         DataGridViewElementStates.Resizable)
   If Pos = -1 Then Exit Do
   MsgBox(Pos)
Loop
```

لاحظ أننا بدأنا البحث من موضع يساوي عدد الصفوف، رغم أن ترقيم الصفوف يبدأ من الصفر وينتهي عند عدد الصفوف -١.. السبب في هذا هو أن هناك صفا زائدا (هو الصف الجديد الذي تجاوره العلامة * في جدول العرض)، لهذا يمكن أن نأخذه في اعتبارنا.. أما لو حاولت أن تبدأ البحث من موضع يزيد على عدد الصفوف (مثل Count + 1) فسيحدث خطأ في البرامج.

- **معرفة عدد الصفوف GetRowCount:**

تعيد عدد صفوف الجدول التي لها الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates.. والمثال التالي يعرض عدد الصفوف المحددة في جدول العرض:

```
MsgBox(DataGridView1.Rows.GetRowCount(
        DataGridViewElementStates.Selected))
```

- **معرفة ارتفاع الصفوف GetRowsHeight:**

تعيد مجموع ارتفاع الصفوف التي لها الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates.

- **صف مشترك SharedRow:**

تعيد كائن الصف DataGridViewRow الذي يمثل الصف المشترك الموجود في الموضع المرسل كمعامل.. وسنتعرف على مفهوم الصفوف المشتركة Shared Rows بالتفصيل لاحقا.

كما تمتلك هذه المجموعة الحدث التالي:

- **المجموعة تغيرت CollectionChanged:**

ينطلق هذا الحدث عند حدوث تغير في عناصر المجموعة بالحذف أو الإضافة.. والمعامل الثاني e لهذا الحدث من النوع CollectionChangeEventArgs وقد تعرفنا عليه من قبل عند التعرف على مجموعة الجداول DataTableCollection.

## فئة صف جدول العرض DataGridViewRow Class

هذه الفئة ترث الفئة DataGridViewBand، وهي تمثل أحد صفوف جدول العرض.. وتمتلك هذه الفئة الخصائص التالية:

### كائن سهولة الوصول AccessibilityObject:

تعيد كائن تسهيل الوصول AccessibleObject المستخدم مع الصف الحالي لتسهيل تعامل ذوي الاحتياجات الخاصة (كضعاف البصر) مع بيانات هذا الصف.. هذا الموضوع خارج نطاق هذا الكتاب.

### الخانات Cells:

تعيد مجموع الخانات DataGridViewCellCollection التي تحتوي على خانات الصف الحالي.. والمثال التالي يعرض قيمة الخانة الأولى في الصف الأول، مع ملاحظة أن صف رءوس الأعمدة، وعمود رءوس الصفوف لا يدخلان في الترقيم:

```
MsgBox(DataGridView1.Rows(0).Cells(0).Value)
```

### العنصر المرتبط بالبيانات DataBoundItem:

تعيد الكائن الذي يعرض الصف الحالي بياناته.. لو أخذت المشروع BindGridToArray كمثال، فإن هذه الخاصية تعيد كائن التلميذ Student الذي يعرضه الصف من المصفوفة Std.. ولو حددت أي صف في الجدول وضغطت الزر DataBoundItem في هذا المشروع، فستظهر لك رسالة تخبرك بتفاصيل كائن التلميذ المرتبط بهذا الصف.

أما لو كان جدول العرض مرتبطا بجدول من مجموعة البيانات، فستعيد هذه الخاصية كائن عرض الصف DataRowView الذي يحوي بيانات الصف المعروض حاليا.

لاحظ أن أي تغيير ستجريه على الكائن المرتبط بالصف الحالي سينتقل إلى مصدر البيانات، ومن ثم سيظهر هذا التغيير في جدول العرض مباشرة.

### الارتفاع Height:

تقرأ أو تغير ارتفاع الصف الحالي.. والقيمة الافتراضية لهذه الخاصية هي الارتفاع المناسب لخط الكتابة الحالي + ٩ نقاط 9 Pixels.

### أقل ارتفاع MinimumHeight:

تقرأ أو تغير أقل ارتفاع يمكن أن يقبله الصف الحالي.. وأقل قيمة لهذه الخاصية هي ٢، والقيمة الافتراضية لها هي ٣.

### ارتفاع الفاصل DividerHeight:

تحدد حجم المساحة التي تفصل الصف الحالي عن الصف التالي.. هذا الفاصل هو مساحة خالية تأخذ نفس لون أرضية جدول العرض، ورغم أنها تعتبر جزءا من الصف الحالي، إلا أنها لا تؤدي أية وظيفة من وظائفه، ما عدا العمل كفاصل شكلي.. والصورة التالية توضح تأثير وضع القيمة ١٠ في هذه الخاصية في الصف الثاني:

| About | Phone | CountryID | Author | ID |
|---|---|---|---|---|
| .... | | ٢١ | توفيق الحكيم | ١٢ |
| ... | | ٢١ | عباس العقاد | ١٣ |
| شاعر مصري معاصر | | ٢١ | فاروق جويدة | ١٤ |
| | | | | |

والقيمة الافتراضية لهذه الخاصية هي صفر، لهذا لا يوجد أي فاصل بين الصفوف، ما عدا خطوط الشبكة العادية.

### نص الخطأ ErrorText:

تقرأ أو تغير النص الذي يشرح الأخطاء التي حدثت في العمود، مثلما يحدث عندما يكتب المستخدم قيما غير مسموح بها في خانات العمود، أو عندما توضع قيمة في الخاصية ErrorText الخاصة بأحد صفوف مجموعة البيانات المرتبطة بجدول العرض، فهذا الخطأ ينتقل آليا إلى جدول العرض.
وتظهر أيقونة حمراء في هامش الصف لتنبه المستخدم إلى وجود خطأ، وعندما يحلق فوقها بالفأرة، سيظهر له تلميح يعرض النص الموجود في الخاصية ErrorText.

### خانة رأس الصف HeaderCell:

تقرأ أو تغير كائن الخانة الرئيسية DataGridViewRowHeaderCell التي تتحكم في خانة رأس الصف الحالي.

**هل هو جديد IsNewRow:**

تعيد True إذا كان الصف الحالي هو الصف الجديد الموجود في آخر صف في جدول العرض وتجاوره العلامة *.. هذا الصف موجود فعلا في مجموعة صفوف الجدول، لكنه يظل صفا جديدا إلى أن يكتب فيه المستخدم أية قيمة، حيث يصبح صفا عاديا، ويضاف صف جديد بدلا منه.

كما تمتلك هذه الفئة الوسائل التالية:

**ضبط طراز حافة رأس الصف AdjustRowHeaderBorderStyle:**

تعدل شكل حافة رأس الصف الحالي عندما يبدأ المستخدم تحريره.. وتستقبل هذه الوسيلة المعاملات التالية:

- كائن طراز الحافة المتطور DataGridViewAdvancedBorderStyle الذي سيتم تعديله.
- كائن طراز الحافة المتطور الذي سيستخدم لحفظ التغييرات البينية التي تحدث لرأس الصف.
- معامل منطقي، إذا جعلت قيمته True ستتم إضافة حافة رأسية مفردة إلى رأس الصف.
- معامل منطقي، إذا جعلت قيمته True ستتم إضافة حافة أفقية مفردة إلى رأس الصف.
- معامل منطقي، أرسل إليه True إذا كان الصف الحالي هو أول صف معروض في الجدول.
- معامل منطقي، أرسل إليه True إذا كان الصف الحالي هو آخر صف معروض في الجدول.

وتعيد هذه الوسيلة كائن طراز الحافة المتطور DataGridViewAdvancedBorderStyle الذي يمثل طراز الحافة المعدل.

لاحظ أنك لست مضطرا إلى استخدام هذه الوسيلة يدويا، فجدول العرض يستدعيها تلقائيا لضبط شكل حواف الخانات الرئيسية للصفوف عند تحريرها.

**إنشاء خانات CreateCells:**

تحذف جميع خانات الصف الحالي، وتنشئ بدلا منها خانات جديدة، كل خانة منها مستمدة من قالب الخانة CellTemplate الخاص بالعمود الذي توجد به.. ولهذه الوسيلة صيغتان:

١- الصيغة الأولى تستقبل كائن جدول العرض DataGridView الذي سيتم استخدام قوالب الخانات الخاصة بأعمدته.
٢- والصيغة الثانية تزيد على الصيغة السابقة بمعامل ثان، يستقبل مصفوفة كائنات Object Array، بها القيم التي ستوضع في الخانات الجديدة.

### معرفة رف القائمة الموضعية GetContextMenuStrip:

تعيد كائن رف القائمة الموضعية ContextMenuStrip الخاص بالصف الذي ترسل رقمه إليها كمعامل.

### معرفة نص الخطأ GetErrorText:

تعيد نص الخطأ الخاص بالصف الذي أرسلت رقمه كمعامل.

### معرفة الارتفاع المفضل GetPreferredHeight:

تعيد أنسب ارتفاع للصف، وهي تستقبل المعاملات التالية:

- رقم الصف المراد حساب أنسب ارتفاع له.
- إحدى قيم المرقم DataGridViewAutoSizeRowMode التي توضح كيف سيتم تغيير حجم الصف تلقائيا، وهذه القيم هي:

| | |
|---|---|
| AllCells | يتم تغيير ارتفاع الصف ليناسب محتويات جميع خاناته، بما فيها الخانة الرئيسية. |
| AllCells ExceptHeader | يتم تغيير ارتفاع الصف ليناسب محتويات جميع خاناته، ما عدا الخانة الرئيسية. |
| RowHeader | يتم تغيير ارتفاع الصف ليناسب محتويات الخانة الرئيسية Header. |

- معامل منطقي، إذا جعلت قيمته True، فسيتم تقدير الحجم المناسب على اعتبار أن عرض الصف سيظل ثابتا.. أما إذا جعلته False، فسيؤخذ في الاعتبار أن الأعمدة التي توجد فيها خانات الصف قد تغير حجمها تلقائيا لمراعاة التغير الذي حدث في ارتفاع الخانات.

### معرفة الحالة GetState:

توضح حالة الصف الذي أرسلت إليها رقمه كمعامل، وهي تعيد إحدى قيم المرقم DataGridViewElementStates الذي تعرفنا عليه من قبل.

| ملحوظة: |
|---|
| كل من الوسائل GetContextMenuStrip، GetErrorText، GetPreferredHeight، GetState تبدو عجيبة للغاية، فمعاملها يرشحها بجدارة لأن تكون وسيلة مشتركة Shared لأنها لا تتعلق بالصف الحالي تحديدا، وإنما تتعامل مع أي صف ترسل إليها رقمه، ولكن رغم هذا فهي ليست وسيلة مشتركة، فيا ترى ما هو السبب؟<br>في الحقيقة، هذه الوسائل مصممة للتعامل مع الصف الحالي، لهذا عليك أن ترسل إليها تحديدا رقم الصف الحالي دون غيره.. والكود التالي سيسبب خطأ في البرنامج بسبب إرسال الرقم صفر (رقم أول صف) إلى الوسيلة GetState الخاصة بالصف رقم ٢ (الصف الثالث):<br>**MsgBox(DataGridView1.Rows(2).GetState(0).ToString)**<br>بينما لو جربت الكود التالي فلن تحدث مشكلة!<br>**MsgBox(DataGridView1.Rows(0).GetState(0).ToStrin)**<br>فيا ترى لماذا تم إنشاء هذه الوسائل بهذا الشكل العجيب؟<br>السبب في هذا هو أن الخاصية DataGridViewRow.Index التي تحمل رقم الصف تعيد -١ إذا كان الصف مشتركا Shared Row، وفي مثل هذه الحالة تناط بك مهمة إرسال رقم الصف الفعلي إلى هذه الوسائل!.. وطبعا عليك ألا تستخدم الخاصية Index لأنها لو أعادت -١ وأرسلته إلى هذه الوسائل فسيحدث خطأ في البرنامج!<br>لو شئت رأيي الشخصي، فتقنية مشاركة الصفوف ــ التي سنتعرف عليها بالتفصيل لاحقا ــ تسبب بعض التعقيد في الأمور، رغم أنها توفر الكثير من مساحة الذاكرة في التطبيقات الكبيرة. |

### وضع القيم SetValues:

أرسل إلى هذه الوسيلة مصفوفة كائنات Object Array لوضع القيم التي تحتويها في خانات الصف الحالي.. وتعيد هذه الوسيلة True إذا نجح وضع جميع القيم في جميع الخانات، أما إذا فشل وضع بعض القيم فستعيد False.. وإذا كانت المصفوفة تحتوى على عناصر أكثر من عدد خانات الصف، فإن القيم الزائدة يتم إهمالها وتوضع باقي القيم في خانات الصف، ولكن هذه الوسيلة تعيد False.. أما إذا كانت المصفوفة تحتوي على عناصر أقل من خانات الصف، فسيتم ملء بعض الخانات بالقيم الموجودة، وستترك باقي الخانات كما هي بدون تغيير.

لاحظ أن عليك إرسال مصفوفة كائنات تحديدا، لأنك لو أرسلت مصفوفة قيمية Value-Type Array (مثل مصفوفة من الأعداد الصحيحة) إلى هذه الوسيلة كمعامل، فإنها ستعتبرها قيمة واحدة وتضعها كلها في أول خانة في الصف!.. ولحل هذه المشكلة أمامك طريقتان:

- فإما أن تحول المصفوفة القيمية إلى مصفوفة كائنات، بطريقة مثل:

```
Dim Arr( ) As Integer = {1, 2, 3, 4}
Dim O(Arr.Length - 1) As Object
Arr.CopyTo(O, 0)
Row.SetValues(O)
```

- وإما أن ترسل القيم كمجموعة من المعاملات المفردة إلى هذه الوسيلة بدون وضعها في مصفوفة، وستقوم هي بجمعها في مصفوفة كائنات، مثل:

```
Row.SetValues(1, 2, 3, 4)
```

## فئة خانة جدول العرض DataGridViewCell Class

هذه الفئة ترث الفئة DataGridViewElement، وهي فئـــة أساســية مجــردة Abstract Base Class، تشتق منها كل أنواع خانات جدول العرض.
وتمتلــك هــذه الفئـــة عــدة خــصائص مماثلـــة لخــصائص كـــائن النطـــاق DataGridViewBand وكائن العمــود DataGridViewColumn وكـــائن الصف DataGridViewRow، لهذا لن نكرر شرحها هنا، وهي:

- مثبتة Frozen
- معروضة Displayed
- للقراءة فقط ReadOnly
- قابلة لتغيير الحجم Resizable
- محددة Selected
- مرئية Visible
- نوع القيمة ValueType
- الوسم Tag
- نص تلميح الأداة ToolTipText
- نص الخطأ ErrorText
- الطراز Style
- لها طراز HasStyle
- الطراز الموروث InheritedStyle
- كائن تسهيل الوصول AccessibilityObject
- رف القائمة الموضعية ContextMenuStrip

وإضافة إلى هذه الخصائص، تمتلك هذه الفئة الخصائص التالية:

### رقم العمود ColumnIndex:
تعيد رقم العمود الذي توجد فيه الخانة الحالية.

### العمود المالك OwningColumn:
تعيد كائن العمود DataGridViewColumn الذي توجد فيه الخانة الحالية.

### رقم الصف RowIndex:
تعيد رقم الصف الذي توجد فيه الخانة الحالية.

### الصف المالك OwningRow:
تعيد كائن الصف DataGridViewRow الذي توجد فيه الخانة الحالية.

### الحجم Size:
تعيد كائن الحجم Size، الذي يحمل أبعاد الخانة الحالية.

### الحجم المفضل PreferredSize:
تعيد كائن الحجم Size، الذي يحمل أنسب أبعاد للخانة الحالية لكي تستوعب محتوياتها استيعابا كاملا.

### حدود المحتوى ContentBounds:
تعيد كائن المستطيل Rectangle، الذي يحمل موضع ومساحة محتويات الخانة الحالية.. تذكر أن الخانة قد تحتوي على نص أو صورة أو مربع اختيار أو زر أو أية أداة أخرى، لهذا فإن هذه الخاصية تعيد إليك حدود الأداة المحتواة في الخانة.

### حدود أيقونة الخطأ ErrorIconBounds:
تعيد كائن المستطيل Rectangle، الذي يحمل موضع ومساحة أيقونة الخطأ التي تعرضها الخانة الحالية (إن وجدت).

### القيمة الافتراضية للصف الجديد DefaultNewRowValue:
تعيد كائنا Object، يحتوي على القيمة الافتراضية للخانة الحالية إذا كانت في الصف الجديد (آخر صف في جدول العرض).
وتفيدك هذه الخاصية عندما تنشئ نوعا خاصا بك من الخانات، ففي هذه الحالة عليك أن تجعل هذه الخاصة تعيد القيمة الافتراضية للخانة الجديدة.. مثلا: لو كنت تتعامل مع خانة تستقبل أيقونات، يمكنك أن تعرض في الخانة الجديدة أيقونة في شكل علامة استفهام.. وإن كنت تتعامل مع خانة تاريخ، فيمكنك أن تضع في الخانة الجديد التاريخ اليوم الحالي.. وهكذا.

### القيمة Value:
تقرأ أو تغير قيمة الخانة الحالية، وهي من النوع Object.. مثال:

```
DataGridView1.Rows(0).Cells(0).Value = "Test"
```

### القيمة المنسقة FormattedValue:
تعيد كائنا Object يحمل القيمة المنسقة للخانة الحالية.. لاحظ أن القيمة الموجودة في الخانة قد تكون نصا مثلا، بينما يتم تنسيق هذا النص كتاريخ.

### نوع القيمة المنسقة FormattedValueType:
تعيد كائن النوع Type الذي يمثل نوع القيمة المنسقة الموجودة في الخانة.

**القيمة المعدلة المنسقة EditedFormattedValue:**

تعيد القيمة الحالية للخانة بالتنسيق المطلوب، حتى لو كانت الخانة في وضع التحرير وكتب بها المستخدم قيمة لم تقبل بعد.. وبهذا تختلف عن الوسيلة FormattedValue، التي تعيد القيمة المحفوظة في الخانة فعلا.

**نوع التحرير EditType:**

تعيد كائن النوع Type، الذي يمثل الأداة التي توضع في الخانة عند تحريرها.. مثلا: عند تحرير أي خانة في عمود نصي، يوضع فيها مربع نص من النوع DataGridViewTextBoxEditingControl لاستقبال ما يكتبه المستخدم.. وعند تحرير خانة في عمود قوائم مركبة، يوضع فيها قائمة مركبة من النوع DataGridViewComboBoxEditingControl.

وقد استبدلنا Override هذه الخاصية في الفئة CalendarCell في المشروع DataGridColumnTypes لجعلها تعيد النوع CalendarEditingControl، وهو نوع أداة تحرير خاصة بنا، أنشأناها للتعامل مع الخانات التي تعرض أداة اختيار التاريخ DateTimePicker.

**هل هو في وضع التحرير IsInEditMode:**

تعيد True إذا كانت الخانة في وضع التحرير حاليا.

**الحالة الموروثة InheritedState:**

تعيد إحدى قيم المرقم DataGridViewElementStates التي توضح حالة الخانة الحالية.

كما تمتلك هذه الفئة الوسائل التالية:

**قياس ارتفاع النص MeasureTextHeight:**

تحسب الارتفاع اللازم لرسم النص المرسل إليها.. هذا مفيد لمعرفة أنسب ارتفاع للصف يناسب هذا النص قبل وضعه في الخانة.. وتستقبل هذه الوسيلة المعاملات التالية:

- كائن الرسوم Graphics المستخدم في رسم النص.
- النص المراد قياس عرضه.
- كائن الخط Font الذي سيكتب النص به.
- أقصى عرض يمكن رسم النص فيه (عرض العمود).
- إحدى قيم المرقم TextFormatFlags توضح تنسيق النص، وهي:

| | |
|---|---|
| Default | التنسيق الافتراضي. |
| Bottom | محاذاة النص إلى أسفل. |
| Top | محاذاة النص إلى أعلى. |
| Left | محاذاة النص إلى اليسار. |
| Right | محاذاة النص إلى اليمين. |
| RightToLeft | رسم النص من اليمين إلى اليسار. |
| HorizontalCenter | توسيط النص أفقيا. |
| VerticalCenter | توسيط النص رأسيا. |
| EndEllipsis | رسم نهايات الخطوط الحادة بحواف مستديرة. |
| PathEllipsis | رسم المسارات الحادة بحواف مستديرة. |
| ExpandTabs | توسيع حروف الجدولة. |
| ExternalLeading | إضافة عرض المسافة البادئة للخط إلى ارتفاع النص. |
| HidePrefix | إخفاء البادئة. |
| Internal | استخدام خط النظام في عملية القياس. |
| ModifyString | ليس لها تأثير. |
| NoClipping | عدم قص علامات التشكيل الفوقية والسفلية التي تتجاوز مستطيل رسم النص. |
| GlyphOverhang Padding | إضافة هامش إلى مستطيل الرسم لاستيعاب علامات التشكيل الزائدة. |
| NoPadding | لا تضاف أية هوامش. |
| LeftAndRight Padding | إضافة هامش أيمن وهامش أيسر فقط. |
| NoPrefix | تجاهل الرمز & الدال على الحروف التذكيرية Mnemonic Characters واعتباره حرفا عاديا. |
| NoFullWidth CharacterBreak | عدم استخدام الكشيدة لمط الحروف لجعل النص يشغل عرض المستطيل بالكامل. |
| PrefixOnly | تستخدم مع ويندوز ٢٠٠٠ و XP. |
| SingleLine | رسم النص في سطر واحد. |
| TextBoxControl | تنسيق النص لعرضه في مربع نص. |

| | |
|---|---|
| WordBreak | تقسيم النص إلى سطور في نهاية الكلمات. |
| WordEllipsis | حذف الكلمات الزائدة عن عرض السطر، ووضع نقاط تكملة (...) بعد آخر كلمة ظاهرة. |
| PreserveGraphics Clipping | استخدام التقطيع الخاص بكائن الرسوم. |
| PreserveGraphics TranslateTransform | استخدام التحويل الإحداثي (كالتكبير والتصغير) الخاص بكائن الرسوم. |

ويمكنك دمج أكثر من قيمة من هذه القيم باستخدام المعامل Or.

وتوجد صيغة أخرى لهذه الوسيلة، تزيد بمعامل سادس على الصيغة السابقة، وهو معامل منطقي مرجعي ByRef يعمل كمعامل إخراج Output، وهو يعيد True إذا كان عرض النص المرسوم أكبر من أقصى عرض مسموح به في المعامل الرابع.

### قياس عرض النص MeasureTextWidth:

تحسب العرض اللازم لرسم النص المرسل إليها، ولها المعاملات التالية:

- كائن الرسوم Graphics المستخدم في رسم النص.
- النص المراد قياس عرضه.
- كائن الخط Font الذي سيكتب النص به.
- أقصى ارتفاع يمكن رسم النص فيه.
- إحدى قيم المرقم TextFormatFlags التي توضح تنسيق النص.

### قياس حجم النص MeasureTextSize:

تعيد كائن حجم Size يحمل العرض والارتفاع اللازمين لرسم النص المرسل إليها، وهي تستقبل المعاملات التالية:

- كائن الرسوم Graphics المستخدم في رسم النص.
- النص المراد قياس عرضه.
- كائن الخط Font الذي سيكتب النص به.
- إحدى قيم المرقم TextFormatFlags التي توضح تنسيق النص.

**قياس الحجم المفضل للنص MeasureTextPreferredSize:**

تعيد كائن حجم Size يحمل أفضل عرض وارتفاع مناسبين لرسم النص المرسل إليها، وهي تستقبل المعاملات التالية:

- كائن الرسوم Graphics المستخدم في رسم النص.
- النص المراد قياس عرضه.
- كائن الخط Font الذي سيكتب النص به.
- عدد مفرد Single أكبر من صفر وأقل من ١، يحمل أقصى نسبة مسموح بها بين عرض وارتفاع النص عند رسمه.
- إحدى قيم المرقم TextFormatFlags التي توضح تنسيق النص.

**ضبط شكل حافة الخانة AdjustCellBorderStyle:**

مماثلة للوسيلة DataGridViewRow.AdjustRowHeaderBorderStyle، ولها نفس المعاملات، ولكنها تقوم بضبط شكل حواف الخانة الحالية عندما تكون في وضع التحرير.

**تجهيز أداة التحرير InitializeEditingControl:**

يقوم جدول العرض باستدعاء هذه الوسيلة مرة واحدة لإضافة أداة التحرير الخاصة بالخانة الحالية إلى أدوات التحرير التي يستخدمها.. وتستقبل هذه الوسيلة المعاملات التالية:

- رقم الصف الذي توجد به الخانة.
- كائن يحتوي على القيمة المنسقة Formatted Value التي ستوضع مبدئيا في أداة التحرير عندما تظهر في الخانة.
- كائن طراز الخانة DataGridViewCellStyle الذي يحتوي على الخصائص الشكلية للخانة، لاستخدامه في جعل أداة التحرير شبيهة بالخانة.

وقد استبدلنا Override هذه الوسيلة في كود الفئة CalendarCell في المشروع DataGridColumnTypes لجعل أداة اختيار التاريخ تعرض نفس التاريخ الموجود في الخانة عند ظهورها لأول مرة.

### تحديد موضع أداة التحرير PositionEditingControl:

تحدد موضع وأبعاد أداة التحرير في جدول العرض، ولها المعاملات التالية:

- معامل منطقي Boolean، إذا جعلت قيمته True فستوضع الأداة في الموضع الذي تحدده باقي المعاملات، إما إذا جعلتها False فسيترك للأداة تحديد موضعها بنفسها.
- معامل منطقي Boolean، إذا جعلت قيمته True فسيتم تحديد حجم الأداة تبعا لباقي المعاملات، إما إذا جعلتها False فسيترك للأداة تحديد حجمها بنفسها.
- كائن مستطيل Rectangle يحمل موضع وأبعاد أداة التحرير.
- كائن مستطيل Rectangle يحمل موضع وأبعاد المساحة التي يجب ألا تتجاوزها أداة التحرير.
- كائن DataGridViewCellStyle، يحمل طراز الخانة.
- معامل منطقي، إذا جعلت قيمته True فستضاف حافة رأسية مفردة إلى رأس الصف.
- معامل منطقي، إذا جعلت قيمته True فستضاف حافة أفقية مفردة إلى رأس الصف.
- معامل منطقي، أرسل إليه True إذا كانت الخانة موجودة في أول عمود يعرضه الجدول.
- معامل منطقي، أرسل إليه True إذا كان كانت الخانة موجودة في أول صف يعرضه الجدول.

### تحديد موضع لوحة التحرير PositionEditingPanel:

مشابهة للوسيلة السابقة، إلا أنها تحدد موضع اللوحة التي توضع عليها أداة التحرير داخل الخانة.. ولهذه الوسيلة نفس معاملات الوسيلة السابقة ما عدا أول معاملين فهما غير موجودين هنا.. كما أن هذه الوسيلة تعيد كائن المستطيل Rectangle الذي يحدد موضع وأبعاد أداة التحرير داخل لوحة التحرير.

### إبعاد أداة التحرير DetachEditingControl:

تزيل أداة الكتابة من الخانة الحالية، وتنهي وضع التحرير.

- **معرفة الحالة الموروثة GetInheritedState:**

تعيد إحدى قيم المرقم DataGridViewElementStates التي توضح الحالة الموروثة من جدول العرض أو العمود أو الصف الذي توجد به الخانة الحالية.. وتستقبل هذه الوسيلة كمعامل رقم الصف الذي توجد به الخانة.

| **ملحوظة:** |
|---|
| لا تستطيع استخدام الخاصية InheritedState مع خانة موجودة في صف مشترك Shared Row لأن الخاصية Index التي تشير إلى موضع هذا الصف تعيد القيمة -١، لهذا يمكنك استخدام الوسيلة GetInheritedState بدلا منها، على أن ترسل إليها رقم الصف الفعلي بنفسك.. لاحظ أن هذا هو الحال نفسه في كل الوسائل التالية التي تستقبل رقم الصف الذي توجد به الخانة كمعامل، فهي مخصصة للتعامل مع الخانات الموجودة في صفوف مشتركة. |

- **معرفة حدود المحتوى GetContentBounds:**

تعيد كائن المستطيل Rectangle، الذي يمثل موضع وأبعاد محتويات الخانة الحالية.. ويجب أن ترسل إلى هذه الوسيلة رقم الصف الذي توجد به الخانة.

- **معرفة القيمة المعدلة المنسقة GetEditedFormattedValue:**

تعيد القيمة المكتوبة حاليا في الخانة حتى ولو كانت في وضع التحرير، وهي تستقبل المعاملين التاليين:

- رقم الصف الذي توجد فيه الخانة.
- قيمة من قيم المرقم DataGridViewDataErrorContexts توضح محتوى الخطأ الخاص بالخانة، وهذه القيم هي:

| | |
|---|---|
| Formatting | خطأ في تنسيق قيمة الخانة. |
| Display | خطأ في عرض القيمة من مصدر البيانات. |
| PreferredSize | خطأ في حساب أفضل حجم للخانة.. فشلت الخانة في تنسيق محتوياتها. |
| RowDeletion | خطأ في حذف أحد الصفوف.. يحدث هذا إذا كان الصف مرتبطا بمصدر بيانات، وأطلق مصدر البيانات خطأ عند محاولة حذف هذا الصف منه. |
| Parsing | خطأ في تحويل البيانات التي كتبها المستخدم أو أنت من مصدر البيانات. |

| | |
|---|---|
| Commit | خطأ في حفظ بيانات الخانة في مصدر البيانات. |
| InitialValue Restoration | خطأ في استعادة القيمة الأصلية للخانة عند محاولة إلغاء التحرير الحالي، وذلك بسبب تغير تنسيق الخانة. |
| LeaveControl | خطأ عند مغادرة جدول العرض، بسبب عدم قدرته على حفظ التغييرات في مصدر البيانات. |
| CurrentCell Change | خطأ عند محاولة مغادرة الخانة الحالية، بسبب وجود أخطاء فيها. |
| Scroll | خطأ في الانزلاق، بسبب ظهور خانة بها مشكلة. |
| Clipboard Content | خطأ عند نسخ محتويات الخانة إلى لوحة القصاصات Clipboard، بسبب عدم إمكانية تحويل محتويات هذه الخانة إلى نص. |

لاحظ أنك تستطيع دمج أكثر من قيمة من قيم هذا المرقم معا باستخدام المعامل Or.

### معرفة رف القائمة الموضعية الموروث GetInheritedContextMenuStrip:

تعيد كائن رف القائمة الموضعية ContextMenuStrip الموروث من جدول العرض أو العمود أو الصف الذي توجد به الخانة الحالية.. وتستقبل هذه الوسيلة كمعامل رقم الصف الذي توجد به الخانة.

### معرفة الطراز الموروث GetInheritedStyle:

تعيد كائن طراز الخانة DataGridViewCellStyle الموروث من الجدول أو العمود الذي توجد به الخانة.. وتستقبل هذه الوسيلة المعاملات التالية:

- كائن طراز الخانة DataGridViewCellStyle الذي ستوضع خصائص الطراز الموروث فيه.
- رقم الصف الذي توجد فيه الخانة.
- معامل منطقي إذا جعلته True، فستضاف خصائص الألوان ضمن الطراز الموروث.

### هل يبدأ الزر وضع التحرير KeyEntersEditMode:

تعيد True إذا كان ضغط الحرف المرسل إليها كمعامل يبدأ تحرير الخانة الحالية، علما بأن معامل هذه الوسيلة من نوع الفئة KeyEventArgs، وهي نوع المعامل e في حدث ضغط الزر KeyPress.. والمثال التالي يخبرك إن كان ضغط حرف الإلغاء Escape يبدأ عملية التحرير أم لا (بافتراض أن Cell هو متغير يشير إلى خانة من خانات الجدول):

```
Dim K As New KeyEventArgs(Keys.Escape)
MsgBox(Cell.KeyEntersEditMode(K)) ' Fales
```

### تحويل القيمة المنسقة ParseFormattedValue:

تحول القيمة المنسقة المعروضة في الخانة إلى القيمة الأصلية، وهي تستقبل المعاملات التالية:

- كائن Object يحمل القيمة المنسقة.
- كائن طراز الخانة DataGridViewCellStyle.
- كائن من النوع "محول القيمة" TypeConverter لاستخدامه لتحويل القيمة المنسقة.. ويمكنك إرسال القيمة Nothing لاستخدام المحول الافتراضي.
- كائن من النوع "محول القيمة" TypeConverter لاستخدامه للتحويل إلى القيمة الأصلية للخانة.. ويمكنك إرسال القيمة Nothing لاستخدام المحول الافتراضي.

لاحظ أن جدول العرض يستدعي هذه الوسيلة تلقائيا بعد إدخال المستخدم لقيمة جديدة في أي خانة، لهذا لست مضطرا إلى استدعائها بنفسك.

## فئة خانة مربع النص
## DataGridViewTextBoxCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي تعمل كخانة في جدول العرض تعرض نصا، وعند تحريرها تعرض مربع نص لاستقبال نص جديد من المستخدم.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخاصية التالية:

### أقصى طول للمدخلات MaxInputLength:
تحدد أقصى عدد من الحروف تقبله الخانة الحالية.

وفي المشروع DataGridColumnTypes، أنشأنا الفئة CalendarCell، وهي ترث الفئة DataGridViewTextBoxCell لتكون قابلة للتحرير، ولكننا استبدلنا بعض خصائصها لتسمح بعرض أداة اختيار التاريخ DateTimePicker بدلا من مربع النص.. ولو فحصت كود هذه الفئة، فستلاحظ أنه بسيط للغاية، فهو يدور حول نوع أداة التحرير المستخدمة في الخانة، ونوع القيمة التي تتعامل معها.

## فئة خانة الزر
## DataGridViewButtonCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي خانة تعرض زرا.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخاصيتين التاليتين:

### طريقة العرض المسطح FlatStyle:

### استخدام نص العمود كقيمة للزر UseColumnTextForButtonValue:
وهما مماثلتان للخاصيتين اللتين تحملان نفس الاسم في الفئة DataGridViewButtonColumn، لكنهما تؤثران فقط على الخانة الحالية، وليس على كل خانات العمود.

# واجـــهة خـــانة التحـــريـــر
# IDataGridViewEditingCell Interface

تقدم هذه الواجهة إلى خانات جدول العرض التي تمثلها، القـــدرة علـــى تحريـــر المستخدم لقيمها.. وتمتلك هذه الواجهة العناصر التالية:

**هل تغيرت قيمة خانة التحرير EditingCellValueChanged:**
اجعل قيمة هذه الخاصية True، إذا تغيرت قيمة الخانة في وضع التحرير.

**القيمة المنسقة لخانة التحرير EditingCellFormattedValue:**
تقرأ أو تغير القيمة المنسقة التي تحتويها الخانة في وضع التحرير.

**معرفة القيمة المنسقة لخانة التحرير GetEditingCellFormattedValue:**
تعيد كائنا Object، يحتوي على القيمة المنسقة لخانة التحرير.. وتـستقبل هذه الوسيلة إحدى قـيم المـرقم DataGridViewDataErrorContexts التي توضح نوع الخطأ الذي حدث بالخانة، وقد تعرفنا عليه من قبل.

**تجهيز خانة التحرير للتحرير PrepareEditingCellForEdit:**
تجهز الخانة عند بدء وضع التحرير، وهي تستقبل معاملا منطقيا، إذا جعلت قيمته True فسيتم تحديد النص المكتوب في الخانة.

## فئة خانة مربع الاختيار
## DataGridViewCheckBoxCell Class

هذه الفئة ترث الفئة DataGridViewCell، كما أنها تمثل الواجهة IDataGridViewEditingCell، وهي تعمل كخانة في جدول العرض تعرض مربع اختيار CheckBox.

ولحدث إنشاء هذه الفئة صيغتان:

١- الصيغة الأولى بدون معاملات.

٢- والصيغة الثانية تستقبل معاملا منطقيا Boolean، يتم إرسال قيمته إلى الخاصية ThreeState.

وإضافة إلى ما ترثه من الفئة الأم، وما تمثله من عناصر الواجهة IDataGridViewEditingCell، تمتلك هذه الفئة بعض الخصائص الشبيهة بخصائص الفئة DataGridViewCheckBoxColumn، وهي تقوم بنفس الوظيفة لكنها تتعامل مع الخانة الحالية فقط وليس العمود كله، لهذا لن نكرر شرحها هنا، وسنكتفي بذكر أسمائها:

- **طريقة العرض المسطح FlatStyle**
- **ثلاثي الحالة ThreeState**
- **القيمة الخاطئة FalseValue**
- **القيمة الصحيحة TrueValue**
- **القيمة غير المحددة IndeterminateValue**

## فئة خانة الصور
## DataGridViewImageCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي تعمل كخانة في جدول العرض تعرض صورة أو أيقونة.

ولحدث إنشاء هذه الفئة صيغتان:

١- الصيغة الأولى بدون معاملات.

٢- والصيغة الثانية تستقبل معاملا منطقيا Boolean، يتم إرسال قيمته إلى الخاصية ValuesAreIcons.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة بعض الخصائص الشبيهة بخصائص الفئة DataGridViewImageColumn، وهي تقوم بنفس الوظيفة لكنها تتعامل مع الخانة الحالية فقط وليس العمود كله، لهذا لن نكرر شرحها هنا، وسنكتفي بذكر أسمائها:

- **الوصف Description**
- **مخطط الصورة ImageLayout**
- **هل القيمة أيقونة ValueIsIcon**

# فئة خانة الوصلة
# DataGridViewLinkCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي تعمل كخانة في جدول العرض، بها وصلة Link.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة بعض الخصائص الشبيهة بخصائص الفئة DataGridViewLinkColumn، وهي تقوم بنفس الوظيفة لكنها تتعامل مع الخانة الحالية فقط وليس العمود كله، لهذا لن نكرر شرحها هنا، وسنكتفي بذكر أسمائها:

- **استخدم نص العمود كقيمة للوصلة UseColumnTextForLinkValue**
- **سلوك الرابط LinkBehavior**
- **لون الرابط LinkColor**
- **لون الرابط الفعال ActiveLinkColor**
- **لون الرابط المُزار VisitedLinkColor**
- **تتبع حالة الزيارة TrackVisitedState**

كما تمتلك هذه الفئة الخاصية التالية:

- **تمت زيارة الرابط LinkVisited:**

إذا جعلت قيمة هذه الخاصية True، فإن الرابط يأخذ اللون المحدد في الخاصية VisitedLinkColor.. لاحظ أن لون الرابط يتغير تلقائيا عندما يضغطه المستخدم إذا كانت للخاصية TrackVisitedState القيمة True.

## فئة خانة القائمة المركبة
## DataGridViewComboBoxCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي تعمل كخانة في جدول العرض بها قامة مركبة ComboBox.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة بعض الخصائص الشبيهة بخصائص الفئة DataGridViewCombokBoxColumn، وهي تقوم بنفس الوظيفة لكنها تتعامل مع الخانة الحالية فقط وليس العمود كله، لهذا لن نكرر شرحها هنا، وسنكتفي بذكر أسمائها:

- **تكملة تلقائية AutoComplete**
- **مصدر البيانات DataSource**
- **عنصر العرض DisplayMember**
- **عنصر القيمة ValueMember**
- **مرتبة Sorted**
- **العناصر Items**
- **طريقة العرض المسطح FlatStyle**
- **طريقة العرض DisplayStyle**
- **طريقة عرض الخانة الحالية فقط DisplayStyleForCurrentCellOnly**
- **أقصى عدد من العناصر المسدلة MaxDropDownItems**
- **عرض القائمة المنسدلة DropDownWidth**

# واجـــــهة أداة التحـــريـــــر

# IDataGridViewEditingControl Interface

تعرّف هذه الواجهة العناصر المشتركة بين أدوات التحرير المستخدمة في خانات جدول العرض، وهي تملك الخصائص التالية:

**جدول العرض EditingControlDataGridView:**

تقرأ أو تغير كائن جدول العرض DataGridView الـذي يعـرض أداة التحرير.

**القيمة المنسقة EditingControlFormattedValue:**

تقرأ أو تغير القيمة المنسقة المعروضة حاليا في أداة التحرير.

**رقم الصف EditingControlRowIndex:**

تقرأ أو تغير رقم الصف الذي تظهر فيه أداة التحرير.

**القيمة تغيرت EditingControlValueChanged:**

اجعل قيمة هذه الخاصية True، إذا تغيرت القيمة المكتوبة في أداة التحرير، عن قيمة الخانة التي تعرض أداة التحرير.

**مؤشر لوحة التحرير EditingPanelCursor:**

تعيد كائنا، يحتوي المؤشر Cursor الذي يتم عرضه عندما تمر الفأرة فوق اللوحة Panel التي تحتوي على أداة التحرير.

**تغييـــر موضـــع أداة التحريـــر عنـــد تغيـــر القيمـــة RepositionEditingControlOnValueChange:**

تعيد True إذا كانت أداة التحرير بحاجة إلى تغيير موضـعها بعـد قيـام المستخدم بالكتابة فيها.. على سبيل المثال: قد يكتب المستخدم نصا طـويلا في أداة التحرير، مما يستلزم أن تقوم بتقسيمه على أكثر من سطر.

كما تمتلك هذه الواجهة الوسائل التالية:

- **تطبيق طراز الخانة على أداة التحرير ApplyCellStyleToEditingControl:**
تجعل لأداة التحرير الخصائص الشكلية الخاصة بكائن طراز الخانة DataGridViewCellStyle الذي ترسله إليها كمعامل.

- **هل تريد أداة التحرير زر الإدخال EditingControlWantsInputKey:**
تعيد True إذا كان الزر المرسل كمعامل هو أحد الأزرار التي تتعامل معها أداة الإدخال، سواء كان حرفا يمكن كتابته، أو وظيفة يمكن أن تؤديها.. أما إذا كانت أداة الإدخال لا تتعامل مع الزر، فإنها تعيد False لتوضح أن على جدول العرض التعامل مع هذا الزر.. ولهذه الوسيلة معاملان:
  - المعامل الأول يستقبل إحدى قيم المرقم Keys التي تعبر عن الزر المضغوط.
  - والمعامل الثاني معامل منطقي، إذا كانت قيمته True فهذا معناه أن جدول العرض يمكنه التعامل مع الزر المضغوط.

- **معرفة القيمة المنسقة GetEditingControlFormattedValue:**
تعيد القيمة المنسقة المكتوبة حاليا في أداة التحرير، وهي تستقبل كمعامل إحدى قيم المرقم DataGridViewDataErrorContexts التي توضح محتوى الخطأ الخاص بالخانة.

- **تجهيز أداة التحرير للتحرير PrepareEditingControlForEdit:**
تجهز الأداة عند بدء وضع التحرير، وهي تستقبل معاملا منطقيا، إذا جعلت قيمته True فسيتم تحديد النص المكتوب في أداة التحرير.

وقد أنشأنا الفئة CalendarEditingControl في المشروع DataGridColumnTypes، وجعلناها تمثل الواجهة IDataGridViewEditingControl لكي تكون أداة تحرير نستطيع استخدامها في جدول العرض.. ولكي نجعلها تعرض أداة اختيار التاريخ في الخانة التي تستضيفها، لم نكتب أكثر من سطر واحد فقط يجعلها ترث الأداة DateTimePicker:

```
Class CalendarEditingControl
  Inherits DateTimePicker
  Implements IDataGridViewEditingControl

  ' الكود الذي يمثل خصائص ووسائل الواجهة
  ' IDataGridViewEditingControl
End Class
```

## فئة أداة تحرير مربع النص
## DataGridViewTextBoxEditingControl

هذه الفئة ترث الفئة TextBox، كما أنها تمثل الواجهة IDataGridViewEditingControl، مما يعني أنها مربع نص مخصص للظهور في خانات جدول العرض عند تحريرها، للسماح للمستخدم بالكتابة في الخانة.
ولا تمتلك هذه الفئة أية عناصر أو خصائص جديدة، غير ما ترثه من الفئة الأم، وما تمثله من عناصر الواجهة IDataGridViewEditingControl.

## فئة أداة تحرير القائمة المركبة
## DataGridViewComboBoxEditingControl Class

هذه الفئة ترث الفئة ComboBox، كما أنها تمثل الواجهة IDataGridViewEditingControl، مما يعني أنها قائمة مركبة مخصصة للظهور في خانات جدول العرض عند تحريرها، للسماح للمستخدم باختيار قيمة منها.
ولا تمتلك هذه الفئة أية عناصر أو خصائص جديدة، غير ما ترثه من الفئة الأم، وما تمثله من عناصر الواجهة IDataGridViewEditingControl.

## فئة الخانة الرئيسية
## DataGridViewHeaderCell Class

هذه الفئة ترث الفئة DataGridViewCell، وهي تعمل كفئة أم تحتوي على الخصائص والوسائل المشتركة بين خانة رأس العمود وخانة رأس الصف.
ولا تمتلك هذه الفئة أية خصائص أو وسائل جديدة غير ما ترثه من الفئة الأم.. قد يبدو لك هذا غريبا، لكن فائدة هذه الفئة تتضح حينما تريد أن تنشئ خانة رئيسية خاصة بك تعرض أيقونة أو مقبضا أو لها شكل خاص، ففي هذه الحالة عليك أن تنشئ فئة ترث الفئة DataGridViewHeaderCell، وتضيف إليها القدرات الجديدة التي تريدها.

## فئة خانة رأس العمود
## DataGridViewColumnHeaderCell

هذه الفئة ترث الفئة DataGridViewHeaderCell، وهي تعمل كخانة تعرض عنوان أحد الأعمدة في جدول العرض.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخاصية التالية:

### اتجاه الترتيب SortGlyphDirection:

تحدد اتجاه الصورة الرمزية Glyph التي يعرضها العمود لتوضح طريقة ترتيبه، وهي تأخذ إحدى قيم المرقم SortOrder التالية:

| | |
|---|---|
| None | العمود غير مرتب (لا تظهر أي صورة). |
| Ascending | العمود مرتب تصاعديا (صورة مثلث رأسه إلى أعلى). |
| Descending | العمود مرتب تنازليا (صورة مثلث رأسه إلى أسفل). |

## فئة الخانة العلوية اليسرى
## DataGridViewTopLeftHeaderCell

هذه الفئة ترث الفئة DataGridViewColumnHeaderCell، وهي تمثل الخانة العلوية اليسرى في جدول العرض، والتي عند ضغطها يتم تحديد كل خانات الجدول.

ولا تمتلك هذه الفئة أية خصائص أو وسائل جديدة غير ما ترثه من الفئة الأم.

## فئة خانة رأس الصف
## DataGridViewRowHeaderCell

هذه الفئة ترث الفئة DataGridViewHeaderCell، وهي تعمل كخانة رأس لأحد صفوف جدول العرض، وعند الضغط عليها يتم تحديد هذا الصف.

ولا تمتلك هذه الفئة أية خصائص أو وسائل جديدة غير ما ترثه من الفئة الأم.

# فئة طراز خانة جدول العرض
# DataGridViewCellStyle Class

هذه الفئة تمثل الواجهة ICloneable، وهي تحمل معلومات التنسيق والشكل الخاص بإحدى خانات جدول العرض.
ولحدث إنشاء هذه الفئة صيغتان:
١- الصيغة الأولى بدون معاملات.
٢-والصيغة الثانية تستقبل كائن طراز خانة DataGridViewCellStyle لاستخدام خصائصه كقيم مبدئية للكائن الحالي.

وتمتلك هذه الفئة بعض الخصائص الشهيرة المألوفة لنا، مثل:

- **الهامش الخارجي Padding**
- **لون الخلفية BackColor**
- **لون الكتابة ForeColor**
- **الخط Font**
- **الوسم Tag**

كما تمتلك هذه الفئة الخصائص التالية:

- **المحاذاة Alignment:**

تحدد كيفية محاذاة محتويات الخانة، وهي تأخذ إحدى قيم المرقم DataGridViewContentAlignment التالية:

| | |
|---|---|
| NotSet | المحاذاة غير محددة. |
| TopLeft | أعلى اليسار. |
| TopCenter | أعلى الوسط. |
| TopRight | يمين الوسط. |
| MiddleLeft | يسار الوسط. |
| MiddleCenter | منتصف الوسط. |
| MiddleRight | يمين الوسط. |
| BottomLeft | أسفل اليسار. |
| BottomCenter | أسفل الوسط. |
| BottomRight | أسفل اليمين. |

**تنسيق Format:**
تستقبل نصا يمثل الصيغة التي ستستخدم لتنسيق محتويات الخانة.. لمزيد من التفاصيل عن صيغ التنسيق، راجع ملاحق كتاب برمجة إطار العمل.

**مزود التنسيق FormatProvider:**
هذه الخاصية من النوع IFormatProvider، وهي تستقبل كائن معلومات الثقافة CultureInfo، الذي يحوي معلومات عن اللغة والمنطقة التي ستستخدم قواعدهما في تنسيق الأرقام والتواريخ والنصوص.. وفي الوضع الافتراضي، تستخدم هذه الخاصية الثقافة المحلية الخاصة بجهاز المستخدم.

**هل لمزود التنسيق القيمة الافتراضية IsFormatProviderDefault:**
تعيد True إذا كانت للخاصية FormatProvider القيمة الافتراضية، وتعيد False إذا كنت وضعت قيمة أخرى في تلك الخاصية.

**القيمة المنعدمة لمصدر البيانات DataSourceNullValue:**
ضع في هذه الخاصية القيمة التي تريد حفظها في مصدر البيانات، إذا ترك المستخدم الخانة الحالية فارغة.. والقيمة الافتراضية لهذه الخاصية هي العدم DbNull.

**هل القيمة المنعدمة لمصدر البيانات افتراضية IsDataSourceNullValueDefault:**
تعيد True إذا كانت للخاصية DataSourceNullValue القيمة الافتراضية، وتعيد False إذا كنت وضعت قيمة أخرى في تلك الخاصية.

**القيمة المنعدمة NullValue:**
ضع في هذه الخاصية كائنا Object يحمل القيمة التي تريد عرضها عندما تكون الخانة فارغة أو تحتوي على القيمة المنعدمة DbNull.. كما أن قيمة هذه الخاصية ستكون هي القيمة الافتراضية للخانات التي تضاف جديدا إلى العمود الذي له الطراز الحالي.. مثال:

```
Gdv.Columns(0).DefaultCellStyle.NullValue = "...."
```

هذا الكود سيجعل النص "..." هو القيمة المنعدمة لخانات العمود الأول في جدول العرض، لهذا سيظهر النص "..." في آخر خانة في هذا العمود (الموجودة في الصف الجديد).

لاحظ أن المستخدم يستطيع وضع القيمة المنعدمة في الخانة أثناء تحريره لها بضغط Ctrl+0 من لوحة المفاتيح.. في هذه الحالة سيتم محو كل محتويات الخانة ووضع قيمة هذه الخاصية فيها.. والقيمة الافتراضية لهذه الخاصية هي نص فارغ "".

**هل القيمة المنعدمة افتراضية IsNullValueDefault:**
تعيد True إذا كانت للخاصية NullValue القيمة الافتراضية، وتعيد False إذا كنت وضعت قيمة أخرى في تلك الخاصية.

**لون خلفية التحديد SelectionBackColor:**
تقرأ أو تغير لون خلفية الخانة المحددة Selected Cell.

**لون النص المحدد SelectionForeColor:**
تقرأ أو تغير لون النص في الخانة المحددة.

**طريقة الالتفاف WrapMode:**
توضح هل سيلتف النص على أكثر من سطر إذا تجاوز عرض الخانة أم لا، وهي تأخذ إحدى قيم المرقم DataGridViewTriState الذي تعرفنا عليه من قبل.

كما تملك هذه الفئة الوسيلة التالية:

**تطبيق الطراز ApplyStyle:**
أرسل إلى هذه الوسيلة كائن طراز الخانة DataGridViewCellStyle الذي تريد نسخ قيم خصائصه إلى خصائص الكائن الحالي.

# فئــة طــراز الحافــة المتطــور
# DataGridViewAdvancedBorderStyle Class

هذه الفئة تمثل الواجهة ICloneable، وهي تحمل معلومات عن شــكل إطــار إحدى خانات جدول العرض.
وتمتلك هذه الفئة الخصائص التالية:

### كل الحواف All:
تحدد طراز جميع حــواف الخانــة، وهــي تأخــذ إحــدى قــيم المــرقم DataGridViewAdvancedCellBorderStyle التالية:

| | |
|---|---|
| NotSet | غير محدد. |
| None | بدون حافة. |
| Single | خط مفرد. |
| Inset | خط غائر. |
| InsetDouble | خط مزدوج غائر. |
| Outset | خط بارز. |
| OutsetDouble | خط مزدوج بارز. |
| OutsetPartial | خط مفرد به جزء مرتفع. |

### الحافة العليا Top:
مماثلة للخاصية السابقة، إلا أنها تتعامل مع الإطار العلوي للخانة.

### الحافة السفلية Bottom:
مماثلة للخاصية السابقة، إلا أنها تتعامل مع الإطار السفلي للخانة.

### الحافة اليسرى Left:
مماثلة للخاصية السابقة، إلا أنها تتعامل مع الإطار الأيسر للخانة.

### الحافة اليمنى Right:
مماثلة للخاصية السابقة، إلا أنها تتعامل مع الإطار الأيمن للخانة.

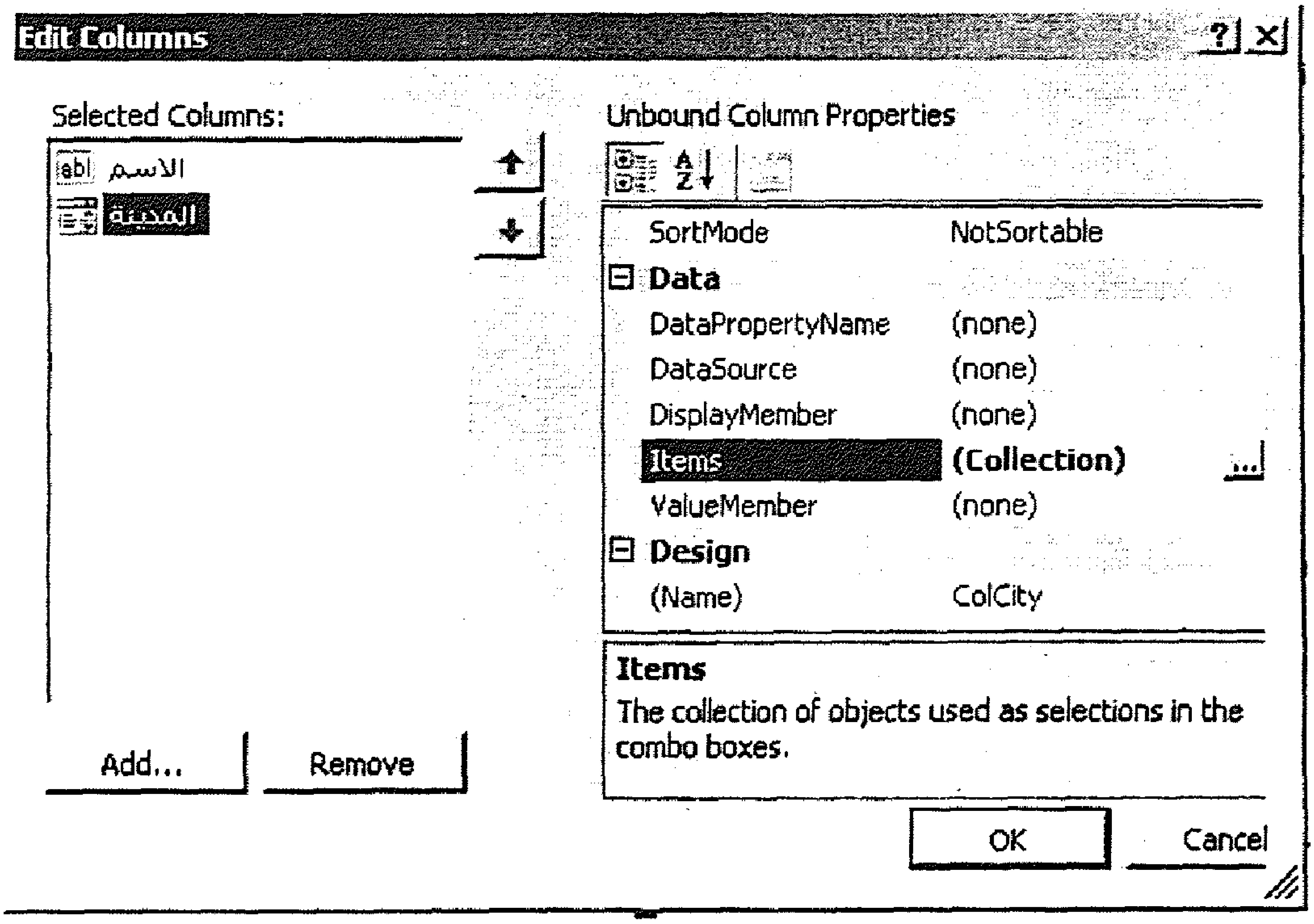
## فئة جدول عرض البيانات DataGridView Class

هذه الفئة ترث فئة الأداة الأم Control Class، وهي تعرض وتتحكم في الأعمدة والصفوف والخانات التي تعرفنا عليها.
ونظرا لأن هذه الأداة تمتلك عددا هائلا من الخصائص والوسائل، فسنقسمها إلـــى مجموعات حسب الوظيفة ليسهل علينا فهمها.

**التعامل مع أعمدة جدول العرض:**
يقدم لك جدول العرض الخصائص التالية، للتعامل مع الأعمدة:

### الأعمدة Columns:

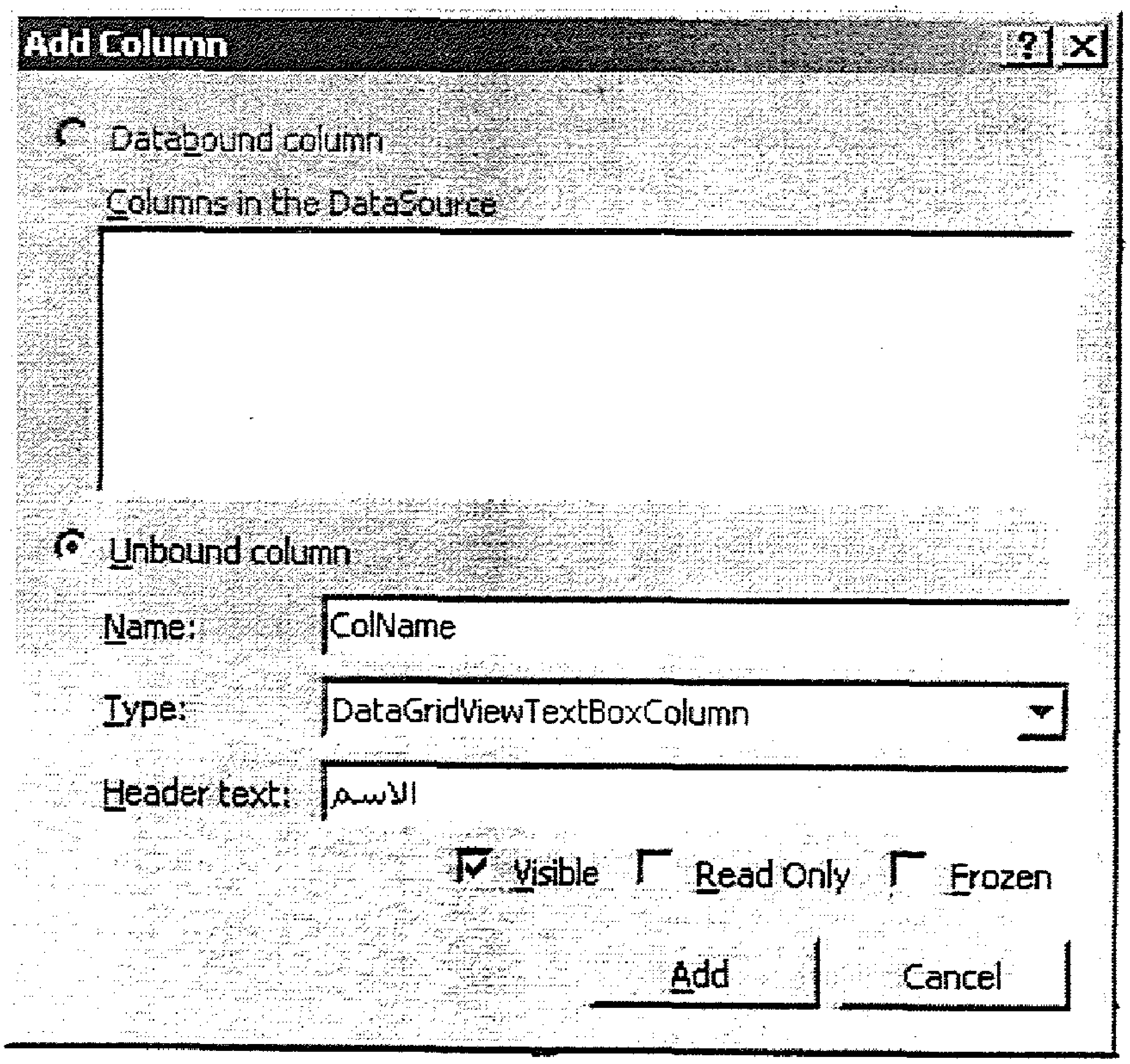
تعيد مجموعة أعمدة جدول العرض DataGridViewColumnCollection، التي تحتوى على كائنات الأعمـــدة DataGridViewColumn Objects الموجودة في جدول العرض.
ويمكنك إضافة الأعمدة إلى هذه المجموعة في وقت التصميم باستخدام نافذة الخصائص، وذلك بضغط زر الانتقال الموجود في خانة قيمة هذه الخاصية، حيث ستظهر لك النافذة الموضحة في الصورة:



في هذه النافذة تظهر أسماء الأعمدة في القائمة اليسرى، بينما تظهر خصائص العمود المحدد في الجزء الأيمن.. على سبيل المثال، تريك الصورة خصائص العمود المسمى "المدينة"، وهو عمود يعرض قائمة منسدلة.. ويمكنك إضافة العناصر إلى هذه القائمة بضغط زر الانتقال الموجود في خانة الخاصية Items في قسم الخصائص، حيث ستظهر لك نافذة تتيح لك إضافة عناصر إلى المجموعة.

ويمكنك حذف أي عمود من القائمة بتحديده وضغط الزر Remove..

ويمكنك إضافة عمود جديد بضغط الزر Add، حيث ستظهر لك نافذة فرعية تتيح لك إدخال بيانات العمود، كما هو موضح في الصورة:




هذه النافذة تتيح لك اختيارين:

١- إنشاء عمود مرتبط بمصدر بيانات Data-bound Column:

هذا الاختيار يكون متاحا فقط إذا كان جدول العرض مرتبطا بمصدر البيانات من خلال الخاصية DataSource.. في هذه الحالة ستعرض القائمة العلوية أسماء الأعمدة المتاحة في مصدر البيانات، وعليك تحديد واحد منها لربط العمود الجديد به.. لاحظ أنك لا تستطيع عرض أعمدة مرتبطة بأكثر من جدول بيانات في نفس جدول العرض.

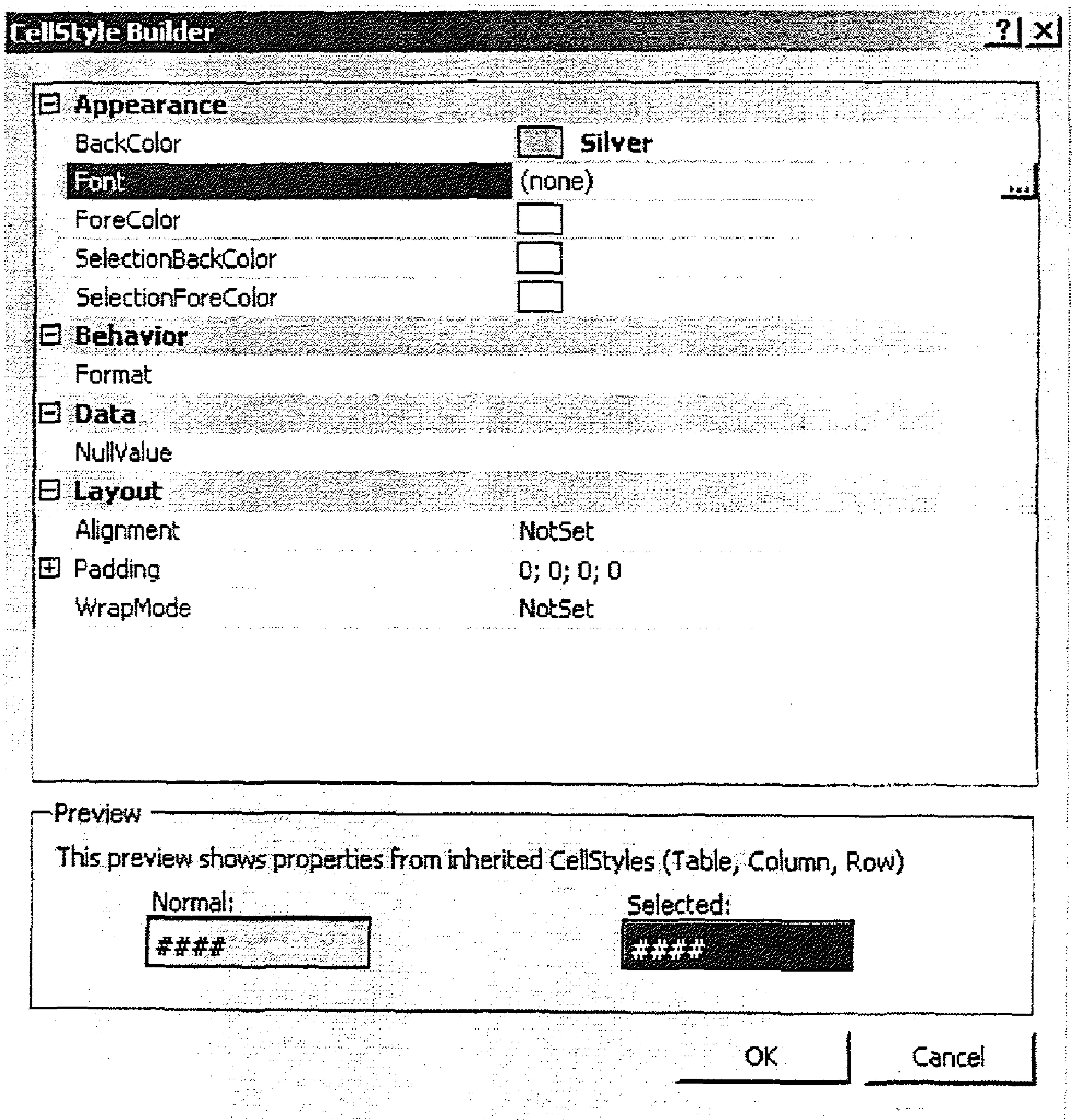
٢- إنشاء عمود غير مرتبط بمصدر بيانات Unbound Column:
هذا الاختيار متاح دائما، ولو فعلته فيجب عليك كتابة تفاصيل العمود كما يلي:

- كتابة الاسم البرمجي للعمود في الخانة Name.. لاحظ أن كل عمود تنشئه في هذه النافذة، يتم تعريف متغير بنفس اسمه على مستوى النموذج، لتستطيع استخدامه مباشرة في التعامل مع العمود، مما يجعل الكود مختصرا.. لهذا عليك اختيار اسم مناسب للعمود يدل على وظيفته، مع وضع بادئة مميزة له (ولتكن Col) كي لا يتعارض مع أي متغيرات أخرى معرفة في البرنامج.. يمكنك مثلا أن تسمي عمود المؤلفين ColAuthors، وعمود الكتب ColBooks.. وهكذا.
- اختيار نوع العمود من القائمة المنسدلة Type.. والعمود النصي DataGridViewTextBoxColumn هو النوع الافتراضي، ويمكنك اختيار أي نوع آخر كعمود القوائم المركبة DataGridViewCombBoxColumn.
- كتابة عنوان العمود (الذي سيعرضه الجدول)، في الخانة Heeder Text.
- إذا كان العمود خفيا لن يظهر للمستخدم، فأزل علامة الاختيار من مربع الاختيار Visible.
- إذا كان العمود للقراءة فقط ولا يمكن للمستخدم تحرير خاناته، فضع علامة الاختيار في مربع الاختيار Read Only.
- إذا كان العمود ثابتا، ولا يمكن للمستخدم تغيير عرضه بالفأرة، فضع علامة الاختيار في مربع الاختيار Frozen.

وبعد أن تنتهي من إدخال تفاصيل العمود، اضغط الزر Add لإضافته إلى مجموعة الأعمدة.. لاحظ أن هذا لن يغلق هذه النافذة، بل سيعيد خاناتها إلى قيمها الافتراضية ليتيح لك إنشاء عمود جديد مباشرة.

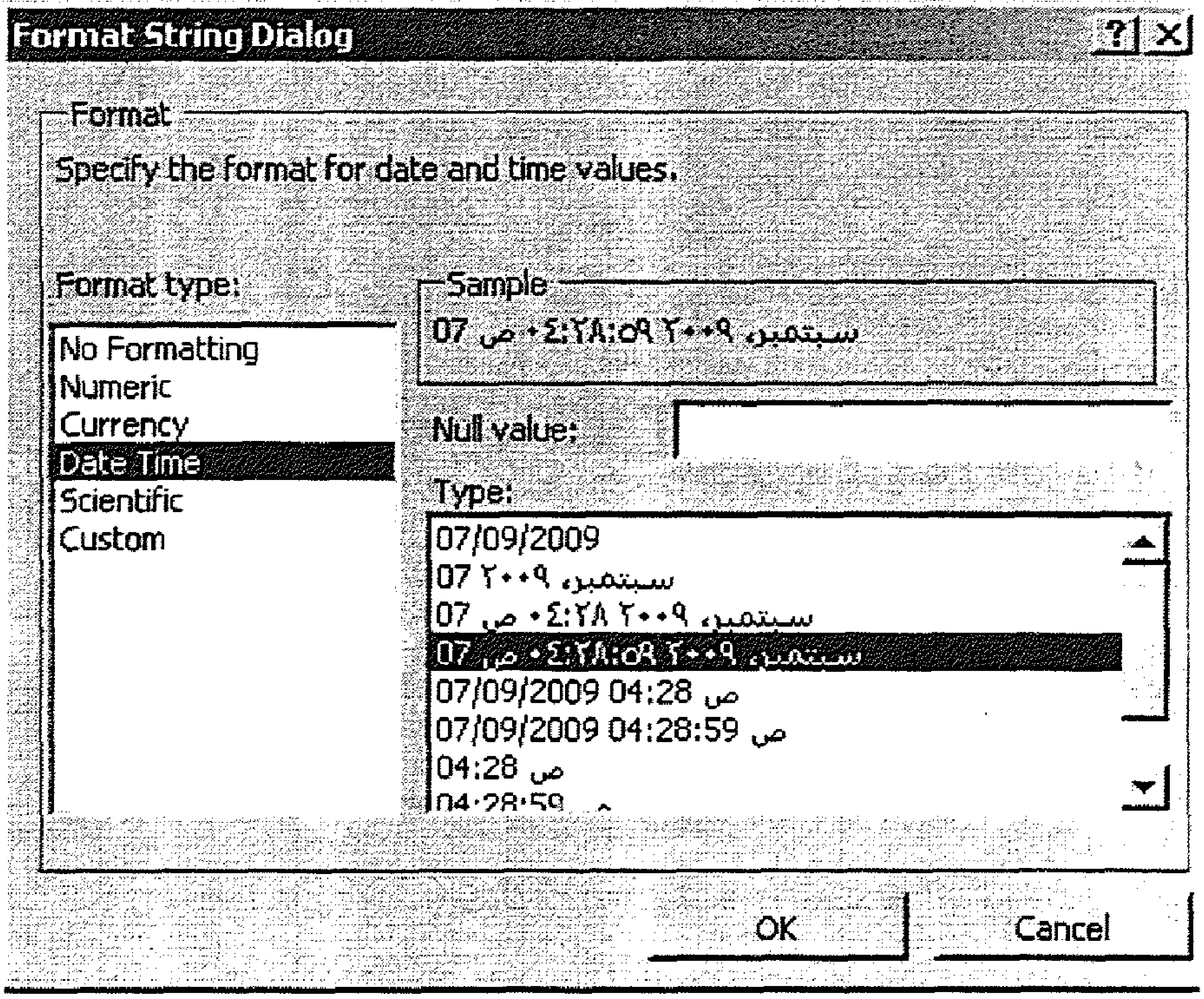
لاحظ أنك تستطيع إنشاء أعمدة مرتبطة وأعمدة غير مرتبطة في نفس الجدول.. لكن هذا سيعقد الأمور عليك، لأن جدول العرض يمحو قيم الأعمدة غير المرتبطة عندما يقوم بتحديث قيم الأعمدة المرتبطة، مثلما يحدث عند ضغط رأس العمود لترتيب الصفوف!.. لهذا بذلنا بعض الجهد في المشروع CustomDataSet للمحافظة على قيم العمود غير المرتبط الذي يعرض أسماء المواد الدراسة في نافذة درجات الطالب.

بعد إن تنتهي من إنشاء كل الأعمدة التي تريدها، اضغط Cancel لإغلاق هذه النافذة والعودة إلى النافذة السابقة، حيث ستجد الأعمدة التي أنشأتها موجودة في قائمة الأعمدة، ومن ثم يمكنك تحديد كل منها وتغيير خصائصه كما تريد.. مثلا، إذا أردت توسيط النص في خانات العمود، فحدد هذا العمود في القسم الأيسر، ومن القسم الأيمن اختر الخاصية DefaultCellStyle، واضغط الزر الموجود في خانة قيمتها، لعرض نافذة باني طراز الخانة CellStyle Builder، التي تتيح لك تغيير خصائص شكل الخانة بصورة مرئية.. كما في الصورة:




هذه النافذة تتيح لك وضع قيم كائن طراز الخانة CellStyle بشكل مرئي وسهل، وهي متاحة للاستخدام أيضا في نافذة الخصائص مع كل الخصائص التي تتعامل مع طراز الخانة، مثل الخاصية DefaultCellStyle.. ويمكنك استخدام هذه النافذة كما يلي:

- اضغط زر الإسدال في خانات خصائص الألوان، لعرض مربع اختيار اللون.
- اضغط زر الإسدال في خانتي المحاذاة Alignment و WrapMode لاختيار القيمة المناسبة من القائمة المنسدلة.
- اضغط زر الانتقال في خانة الخط Font، لعرض مربع اختيار الخط.
- اضغط زر الانتقال في خانة التنسيق Format، لعرض مربع إنشاء نص التنسيق، وهو كما في الصورة:




في هذه النافذة يمكنك اختيار نوع التنسيق من القائمة اليسرى، حيث ستظهر في الجانب الأيمن بعض الاختيارات التي تتيح لك إنشاء صيغة هذا التنسيق، وهي:

- مربع نص القيمة المنعدمة Null Value لتكتب فيه القيمة التي ستستخدم عندما يترك المستخدم الخانة فارغة.
- قائمة تعرض لك صيغ التاريخ المختلفة لتختار منها.
- مربع رقمي NumericUpDown يتيح لك تحديد عدد الخانات العشرية في صيغ الأرقام والعملة والنسب المئوية.

ما يعنيا هنا هو الخاصية Alignment التي تتيح لك اختيار محاذاة النص في خانات العمود.. ولتوسيط النص، اختر القيمة MiddleCentre من القائمة المنسدلة، واضغط Ok للعودة إلى نافذة الأعمدة.

وبعد أن تنتهي من إنشاء كل الأعمدة وضبط خصائصها، اضغط OK لإغلاق النافذة.. ستجد الأعمدة التي أنشأتها قد ظهرت في جدول العرض في وقت التصميم.

### عدد الأعمدة ColumnCount:

تقرأ أو تغير عدد الأعمدة الموجودة في جدول العرض.. لاحظ أن وضع الرقم ٠ في هذه الخاصية سيحذف كل أعمدة الجدول، وإذا جعلت لها قيمة أقل من عدد الأعمدة الحالية، فسيتم حذف بعض الأعمدة من نهاية مجموعة الأعمدة Columns، أما إذا وضعت فيها عددا أكبر من عدد الأعمدة الحالي، فسيتم إنشاء أعمدة نصية وإضافتها إلى نهاية مجموعة الأعمدة، وستكون هذه الأعمدة بدون عناوين.. وتسبب هذه الخاصية خطأ إذا حاولت استخدامها مع جدول عرض مرتبط بمصدر بيانات.

### السماح للمستخدم بترتيب الأعمدة AllowUserToOrderColumns:

إذا جعلت قيمة هذه الخاصية True، فسيستطيع المستخدم سحب الأعمدة من مواضعها لإعادة ترتيبها.. والقيمة الافتراضية لهذه الخاصية False.

### السماح للمستخدم بتغيير حجم الأعمدة AllowUserToResizeColumns:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيستطيع المستخدم سحب الحافة اليمنى للعمود بالفأرة لتكبير عرضه أو تصغيره.

### إنتاج الأعمدة تلقائيا AutoGenerateColumns:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتم إنشاء الأعمدة تلقائيا عند ربط جدول العرض بمصدر البيانات.

### طريقة التحجيم التلقائي للأعمدة AutoSizeColumnsMode:

توضح كيف يتم تغيير عرض الأعمدة تلقائيا تبعا لمحتوياتها، وهي تأخذ إحدى قيم المرقم DataGridViewAutoSizeColumnsMode، وهو يملك نفس قيم المرقم DataGridViewAutoSizeColumnMode الذي تعرفنا عليه سابقا، ما عدا القيمة NotSet.

### رءوس الأعمدة مرئية ColumnHeadersVisible:

إذا جعلت قيمة هذه الخاصية False، فلن يتم عرض الصف الذي يحتوي على رؤوس الأعمدة.. والقيمة الافتراضية لهذه الخاصية هي True.

### ارتفاع رءوس الأعمدة ColumnHeadersHeight:
تقرأ أو تغير ارتفاع الصف الذي يحوي رءوس الأعمدة.

### طريقة تغيير ارتفاع رءوس الأعمدة ColumnHeadersHeightSizeMode:
تحدد كيفية تغير ارتفاع الصف الذي يحتوي على رءوس الأعمدة، وهي تأخذ إحدى قيم المرقم DataGridViewColumnHeadersHeightSizeMode:

| | |
|---|---|
| EnableResizing | السماح للمستخدم بتغيير ارتفاع رءوس الأعمدة. |
| DisableResizing | عدم السماح للمستخدم بتغيير ارتفاع رءوس الأعمدة. |
| AutoSize | تغيير ارتفاع رءوس الأعمدة تلقائيا ليناسب عناوينها. |

### عرض الجزء المختفي من أول عمود ظاهر FirstDisplayedScrollingColumnHiddenWidth:
تعيد عرض الجزء المختفي من أول عمود ظاهر على الشاشة حاليا.

### رقم أول عمود ظاهر FirstDisplayedScrollingColumnIndex:
تعيد رقم أول عمود ظاهر على الشاشة حاليا.. ويمكنك أيضا تغيير قيمة هذه الخاصية، للانزلاق إلى العمود الذي أرسلت رقمه إليها، ليصير أول عمود ظاهر.. والمثال التالي ينزلق إلى العمود العاشر إن لم يكن ظاهرا على الشاشة:

**Dgv.FirstDisplayedScrollingColumnIndex = 9**

ويؤدي استخدام رقم عمود غير موجود إلى حدوث خطأ في البرنامج.. على سبيل المثال، ستسبب الجملة السابقة خطأ إذا كان عدد الأعمدة أقل من ١٠.

### الأعمدة المحددة SelectedColumns:
تعيد مجموعة أعمدة DataGridViewSelectedColumnCollection، وهي مجموعة ترث الفئة BaseCollection وتمثل واجهة القائمة IList، وتحتوي على الأعمدة المحددة حاليا في الجدول.

### عمود الترتيب SortedColumn:
تعيد كائن العمود DataGridViewColumn الذي تم ترتيب صفوف جدول العرض تبعا لترتيب خاناته.. وتعيد هذه الخاصية Nothing إذا لم يكن جدول العرض مرتبا.

**طراز حواف رؤوس الأعمدة ColumnHeadersBorderStyle:**

تحدد شكل إطار خانات العناوين، وهي تأخذ إحدى قيم المرقم DataGridViewHeaderBorderStyle التالية:

| | |
|---|---|
| None | لا توجد إطارات. |
| Single | إطار من خط مفرد. |
| Raised | إطار بارز. |
| Sunken | إطار غائر. |
| Custom | إطار مخصص.. هذه القيمة للقراءة فقط، ولا يمكنك وضعها بنفسك، وإنما تتغير تلقائيا عند تغيير قيمة الخاصية AdvancedColumnHeadersBorderStyle. |

**الطراز الافتراضي لخانات رءوس الأعمدة ColumnHeadersDefaultCellStyle:**

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle المستخدم مع خانات رؤوس الأعمدة.. ويمكنك تغيير هذا الطراز بشكل مرئي من نافذة الخصائص، وذلك بضغط زر الانتقال الموجود في خانة قيمة هذه الخاصية لعرض نافذة باني طراز الخانة CellStyle Builder.

**الطراز المتطور لحافة رؤوس الأعمدة AdvancedColumnHeadersBorderStyle:**

تعيد كائن الطراز المتطور DataGridViewAdvancedBorderStyle الذي يتعامل مع إطارات خانات عناوين الأعمدة.

**طراز الحافة المضبوط للخانة العلوية اليسرى AdjustedTopLeftHeaderBorderStyle:**

تعيد كائن الطراز المتطور DataGridViewAdvancedBorderStyle الذي يتعامل مع إطارات الخانة العلوية اليسرى في الجدول.

**تفعيل الطرازات الشكلية للخانات الرئيسية EnableHeadersVisualStyles:**

إذا جعلت قيمة هذه الخاصية True، فسيتم استخدام طراز الحواف وطراز الخانات مع رءوس الأعمدة ورءوس الصفوف.

كما يمدك جدول العرض بالوسائل التالية للتعامل مع الأعمدة:

- **عدد الأعمدة المعروضة DisplayedColumnCount:**
تعيد عدد الأعمدة التي يراها المستخدم على الشاشة في هذه اللحظة، ولها معامل منطقي، إذا جعلت قيمته True فسيدخل ضمن العدد الأعمدة التي يظهر جزء منها فقط.

- **تعديل طراز حافة عنوان العمود AdjustColumnHeaderBorderStyle:**
تعدل شكل إطار رأس العمود.. وتستقبل هذه الوسيلة المعاملات التالية:
  - كائن طراز الحافة المتطور DataGridViewAdvancedBorderStyle الخاص بالعمود الذي سيتم تعديله.
  - كائن طراز الحافة المتطور DataGridViewAdvancedBorderStyle الذي سيستخدم لحفظ التغييرات البينية التي تحدث لرأس العمود.
  - معامل منطقي، أرسل إليه True إذا كان العمود هو أول عمود معروض في الجدول.
  - معامل منطقي، أرسل إليه True إذا كان العمود هو آخر عمود مرئي في الجدول.

وتعيد هذه الوسيلة كائن طراز الحافة المتطور DataGridViewAdvancedBorderStyle الذي يمثل طراز الحافة المعدل.
لاحظ أنك لست مضطرا إلى استخدام هذه الوسيلة يدويا، فجدول العرض يستدعيها تلقائيا لضبط شكل حواف الخانات الرئيسية للأعمدة عند رسمها.

- **تحجيم العمود تلقائيا AutoResizeColumn:**
تغير عرض العمود المطلوب، ليناسب محتويات خاناته.. ولها صيغتان:
  1. الصيغة الأولى تستقبل رقم العمود المراد تحجيمه.
  2. الصيغة الثانية تزيد على الصيغة السابقة بمعامل ثان، يستقبل إحدى قيم المرقم DataGridViewAutoSizeColumnMode لتوضح طريقة تحجيم العمود، وقد سبق لنا التعرف عليه.

**تحجيم تلقائي للأعمدة AutoResizeColumns:**
تغير عرض جميع أعمدة الجدول لتلائم محتويات خاناتها.. ولها صيغتان:
١. الصيغة الأولى بدون معاملات.
٢. الصيغة الثانية تستقبل إحدى قيم المرقم DataGridViewAutoSizeColumnMode لتوضح طريقة تحجيم الأعمدة.

**تغيير ارتفاع رؤوس الأعمدة تلقائيا AutoResizeColumnHeadersHeight:**
تغير ارتفاع رءوس الأعمدة تلقائيا لتناسب محتوياتها.. ولها صيغتان:
١- الصيغة الأولى بدون معاملات، وهي تغير ارتفاع صف العناوين ليراعي محتويات جميع خاناته.
٢- والصيغة الثانية تستقبل رقم العمود الذي يجب مراعاة محتويات خانة عنوانه عند تغيير ارتفاع الصف.

**معرفة مستطيل عرض العمود GetColumnDisplayRectangle:**
تعيد كائن مستطيل Rectangle يحتوي على موضع وأبعاد العمود المطلوب، وهي تستقبل معاملين:
- رقم العمود المطلوب.
- معامل منطقي إذا جعلت قيمته True فستعيد هذه الوسيلة المستطيل المحيط بالجزء المعروض من العمود على الشاشة، وإذا جعلته False، فستعيد المستطيل المحيط بكامل العمود حتى لو كان جزء منه غير ظاهر على الشاشة.

**إبطال رسم العمود InvalidateColumn:**
أرسل إلى هذه الوسيلة رقم العمود، لتقوم بإبطال رسمه في الجدول، مما يجبره على إعادة رسم نفسه من جديد.

كما يمدك جدول العرض بالأحداث التالية للتعامل مع الأعمدة، علما بأن المعامل الثاني e في معظم هذه الأحداث من النوع DataGridViewColumnEventArgs، وهو يمتلك الخاصية Column التي تعيد كائن العمود DataGridViewColumn الذي سبب انطلاق الحدث.. لهذا لن نكرر ذكر هذا في الأحداث، وسنذكر نوع المعامل فقط إذا كان مختلفا:

**إضافة عمود ColumnAdded:**

ينطلق عند إضافة عمود إلى جدول العرض.

**حذف عمود ColumnRemoved:**

ينطلق عند حذف عمود من جدول العرض.

**تغير رقم عرض العمود ColumnDisplayIndexChanged:**

ينطلق عندما تتغير قيمة الخاصية DisplayIndex الخاصة بأحد أعمدة جدول العرض سواء من الكود، أو بسبب سحب المستخدم للعمود من موضعه.

**النقر المزدوج على فاصل العمود ColumnDividerDoubleClick:**

ينطلق عندما ينقر المستخدم مرتين بالفأرة فوق الخط الفاصل بين عمودين، لتحجيم العمود تلقائيا ليناسب محتويات خاناته.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewColumnDividerDoubleClickEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | ColumnIndex | تعيد رقم العمود الذي نقره المستخدم بالفأرة. |
| | Button | تعيد إحدى قيم المرقم MouseButtons التي تخبرك بزر الفأرة الذي ضغطه المستخدم. |
| | Clicks | تعيد عدد مرات ضغط زر الفأرة. |
| | Delta | تعيد عدد حركات عجلة الفأرة. |
| | X | تعيد الموضع الأفقي لمؤشر الفأرة. |
| | Y | تعيد الموضع الرأسي لمؤشر الفأرة. |
| | Location | تعيد كائن النقطة Point، الذي يحمل موضع مؤشر الفأرة. |
| | Handled | إذا جعلت قيمة هذه الخاصية True، فلن تتخذ أية خطوات إضافية لمعالجة الحدث.. هذا معناه إلغاء عملية التحجيم التلقائي للعمود. |

**ضغط رأس العمود ColumnHeaderMouseClick:**

ينطلق عند الضغط بالفأرة على رأس أحد أعمدة جدول العرض.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellMouseEventArgs، وهو يمتلك نفس خصائص الحدث السابق ما عدا الخاصية Handled، كما يمتلك الخاصية RowIndex التي تعيد رقم الصف الذي توجد بـه الخانـة المضغوطة.. لاحظ أن هذه الخاصية ستعيد -١ في هذا الحدث، لأن صـف رؤوس الأعمدة لا يدخل ضمن ترقيم صفوف الجدول.

**النقر المزدوج على راس العمود ColumnHeaderMouseDoubleClick:**

ينطلق عند النقر مرتين بالفأرة على أحد أعمدة الجدول.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellMouseEventHandler كما في الحدث السابق.

**تغير حالة العمود ColumnStateChanged:**

ينطلق عندما تتغير حالة العمود، كأن يفقد المؤشر Lost Focus.. والمعامل الثاني e من النوع DataGridViewColumnStateChangedEventArgs وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| 🔒 | Column | تعيد كائن العمود DataGridViewColumn الذي تغيرت حالته. |
| 🔒 | StateChanged | تخبرك بالحالة الجديدة للعمود، وهـي تعيـد إحـدى قـيم المـرقم DataGridViewElementStates الـذي تعرفنا عليه من قبل عند التعـرف علـى الخاصية DataGridViewElement.State. |

**تغير عرض العمود ColumnWidthChanged:**

ينطلق عندما تتغير قيمة الخاصية Width الخاصة بأحـد أعمـدة جـدول العرض، سواء برمجيا أو بواسطة المستخدم.

## التعامل مع صفوف جدول عرض البيانات:

يمنحك جدول العرض الخصائص التالية للتعامل مع الصفوف:

### عدد الصفوف RowCount:

تقرأ أو تغير عدد صفوف جدول العرض.. ولو وضعت في هذه الخاصية قيمة أكبر من عدد صفوف الجدول، فسيضيف هذا صفوفا جديدة إلى نهاية الجدول، بينما يؤدي وضع قيمة أصغر من عدد صفوف الجدول إلى حذف صفوف من نهاية الجدول، ولو وضعت في هذه الخاصية القيمة صفر فستمحى كل الصفوف.

### الصفوف Rows:

تعيد مجموعة الصفوف DataGridViewRowCollection التي تحتوي على كائنات صفوف جدول العرض DataGridViewRow Objects.. وكل صف تضيفه إلى هذه المجموعة يظهر في جدول العرض، وكل صف تحذفه منها يختفي من جدول العرض.. وتبدأ الصفوف في هذه المجموعة بالصف رقم صفر، وهو أول صف بعد صف رؤوس الأعمدة، وتنتهي هذه المجموعة بالصف الجديد إذا كان مسموحا بعرضه في جدول العرض، أو بآخر صف حقيقي يحتوي على بيانات إن كان جدول العرض لا يعرض الصف الجديد.

لاحظ أن صف رؤوس الأعمدة هو الصف رقم -١، لكنك لا تستطيع التعامل معه من خلال هذه الخاصية، وإنما تحصل على الرقم -١ من الوسائل والأحداث التي تخبرك برقم الصف الذي حدث له تغير معين، كما سنرى فيما يلي.. وبدلا من هذا ويمكن التعامل مع أي خانة في صف الرءوس باستخدام الخاصية HeaderCell لكل عمود.. والمثال التالي سيخبرك أن خانة رأس العمود الأول توجد في الصف رقم -١:

**MsgBox(DGAuthors.Columns(0).HeaderCell.RowIndex)**

ويريك الزر "عكس التحديد" في المشروع DataGridViewAuthorBooks مثالا على كيفية استخدام المجموعة Rows لعكس تحديد الصفوف جدول العرض،وذلك بالمرور على كل صفوف الجدول، وعكس قيمة الخاصية Selected لكل منها.

### الصفوف المحددة SelectedRows:

تعيد مجموعة الصفوف المحددة DataGridViewSelectedRowCollection، وهي مجموعة ترث الفئة BaseCollection وتمثل واجهة القائمة IList، وتحتوي على الأعمدة المحددة حاليا في الجدول.. لاحظ أنك لا تستطيع إضافة صفوف إلى هذه المجموعة لأنها للقراءة فقط ولا تملك الوسيلة Add.. لهذا لو أردت تحديد أحد الصفوف، فضع القيمة True في الخاصية Selected الخاصة بهذا الصف.. والكود التالي يحدد الصف الأول:

```
Dgv.Rows(0).Selected = True
```

لكن هذا لن يزيل تحديد الصفوف المحددة سابقا، لهذا لو أردت فعل هذا، فعليك المرور عبر كل الصفوف المحددة ووضع القيمة False في الخاصية Selected الخاصة بكل منها:

```
Do Until DGAuthors.SelectedRows.Count = 0
   Dgv.SelectedRows(0).Selected = False
Loop
Dgv.Rows(0).Selected = True
```

لاحظ أن إزالة تحديد الصف الأول يحدث مجموعة الصفوف المحددة لإزالته منها، لهذا نستمر في حذف الصف الأول من هذه المجموعة دائما إلى أن تفرغ نهائيا من محتوياتها.

وهناك طريقة أخرى أكثر كفاءة، وهي وضع مجموعة الصفوف المحددة في متغير، وإزالة تحديد كل عناصرها.. صحيح أن المتغير يشير إلى مجموعة الصفوف المحددة مرجعيا، والمفروض أن يرى التغييرات التي تحدث لها، لكن جدول العرض يلغي المجموعة كلها إذا تغير تحديد أي صف وينشئ مجموعة جديدة ويضف إليها الصفوف المحددة، لهذا يظل المرجع الذي وضعناه في المتغير يشير إلى المجموعة القديمة.. لعل هذا يوضح لك لماذا يكون التعامل مع المجموعة SelectedRows مكلفا جدا إذا كان جدول العرض يحتوي على عدد هائل من الصفوف، فتحديث جدول العرض لهذه المجموعة عملية تتسم بعدم الكفاءة!

وستجد الكود التالي في الزر "تحديد الصف الأول" في المشروع DataGridViewAuthorBooks، وهو يزيل تحديد كل الصفوف المحددة، ثم يحدد الصف الأول في جدول العرض:

```
Dim SelRows = DGAuthors.SelectedRows
For Each R As DataGridViewRow In SelRows
     R.Selected = False
Next
DGAuthors.Rows(0).Selected = True
```

**قالب الصف RowTemplate:**
تقرأ أو تغير كائن الصف DataGridViewRow الذي يستخدم كقالب تستمد منه الصفوف الجديدة خصائصها.. هذا مفيد إذا أردت تغيير شكل كل صفوف الجدول ووضع القيم الافتراضية للصف الجديد، فكل ما عليك هو تعريف كائن صف جديد وضبط خصائصه ثم وضعه في هذه الخاصية.. ويمكنك فعل هذا في وقت التصميم باستخدام نافذة الخصائص، فلو ضغطت العلامة + المجاورة لاسم هذه الخاصية، فستظهر بعض الخصائص الفرعية للصف الذي يعمل كقالب، ما يتيح لك التحكم في القائمة الموضعية للصف وارتفاعه وعرض الفاصل، كما يمنك استخدام الخاصية DefaultCellStyle للتحكم شكل خانات الصف.. وقد فعلنا هذا في جدول العرض الموضوع على النموذج FrmBooks في المشروع DataGridViewAuthorBooks، لجعل لون خلفية الخانات اصفر، ولون الكتابة أحمر.

**السماح للمستخدم بإضافة صفوف AllowUserToAddRows:**
إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيعرض جدول العرض صفا إضافيا فارغا في نهاية الجدول، وعند تحرير المستخدم لأي خانة من خاناته يضاف هذا الصف إلى الجدول، ويضاف بعده صف جديد فارغ.

**رقم الصف الجديد NewRowIndex:**
تعيد رقم الصف الجديد في جدول العرض (آخر صف في الجدول).. لاحظ أن هذه الوسيلة ستعيد -١ إذا لم يكن مسموحا للجدول بعرض صف جديد.

**السماح للمستخدم بحذف الصفوف AllowUserToDeleteRows:**
إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتمكن المستخدم من حذف الصف المحدد في جدول العرض بضغط الزر Delete من لوحة المفاتيح.

**السماح للمستخدم بتحجيم الصفوف AllowUserToResizeRows:**
إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتمكن المستخدم من تغيير ارتفاع الصف بسحب حافته بالفأرة.

### الصف الحالي CurrentRow:
تعيد كائن الصف DataGridViewRow الذي يحتوي على الخانة التي بها المؤشر Focused حاليا.

### طريقة التحجيم التلقائي للصف AutoSizeRowsMode:
تحدد كيف سيتم تغيير ارتفاع صفوف جدول العرض تلقائيا، وهي تأخذ إحدى قيم المرقم DataGridViewAutoSizeRowsMode التالية:

| | |
|---|---|
| None | لا يتم تغيير ارتفاع الصفوف تلقائيا. |
| AllCells | تغيير ارتفاع كل صف ليناسب محتويات جميع خاناته، بما فيها الخانة الرئيسية Header. |
| AllCells ExceptHeader | تغيير ارتفاع كل صف ليناسب محتويات جميع خاناته، ما عدا الخانة الرئيسية. |
| AllHeaders | تغيير ارتفاع كل صف ليناسب محتويات الخانة الرئيسية. |
| DisplayedCells | تغيير ارتفاع كل صف ليناسب محتويات خاناته المعروضة على الشاشة، بما فيها الخانة الرئيسية. |
| DisplayedCells ExceptHeaders | تغيير ارتفاع كل صف ليناسب محتويات خاناته المعروضة على الشاشة، ما عدا الخانة الرئيسية. |
| DisplayedHeaders | تغيير ارتفاع كل صف ليناسب محتويات الخانات الرئيسية المعروضة على الشاشة. |

### هل الصف الحالي قذر IsCurrentRowDirty:
تعيد True إذا كان الصف الحالي يحتوي على تغييرات لم يتم حفظها في مصدر البيانات.. لاحظ أن التغييرات التي أجراها المستخدم يتم حفظها في مصدر البيانات في الحالات التالية:

- فور مغادرة الصف الحالي إلى صف آخر.
- إذا ضغط المستخدم Ctrl+Enter من لوحة المفاتيح وهو ما زال في الصف الحالي.
- إذا قمت باستدعاء الوسيلة Form.Validate الخاصة بالنموذج الذي يوجد عليه جدول العرض.
- إذا كان جدول العرض مرتبطا بمصدر ربط BindingSource واستدعيت الوسيلة EndEdit الخاصة به.

وقد استخدمنا هذه الخاصية في المشروع CustomDataSet في الحدث CellContentClick، وذلك لحفظ بيانات التلميذ الحالي في مجموعة البيانات قبل عرض درجاته، حتى لا يحدث خطأ عند محاولة التعامل معها.. هذا هو الكود الذي يفعل هذا:

```
If DgStudents.IsCurrentRowDirty Then
   DgStudents.EndEdit( ) ' إنهاء التحرير
   ' إجبار جدول العرض على نقل التغييرات إلى مجموعة البيانات
   Me.Validate( )
End If
```

**رقم أول صف معروض FirstDisplayedScrollingRowIndex:**

تعيد رقم أول صف معروض حاليا على الشاشة.. ويمكنك أيضا أن تضع فيها رقم الصف الذي تريد الانزلاق إليه ليصير أول صف معروض.

**هل رؤوس الصفوف مرئية RowHeadersVisible:**

إذا جعلت قيمة هذه الخاصية False، فلن يظهر العمود الذي يحتوي رؤوس صفوف الجدول.. والقيمة الافتراضية هي True.

**عرض رءوس الصفوف RowHeadersWidth:**

تقرأ أو تغير عرض العمود الذي يحتوي رؤوس صفوف الجدول.

**طريقة تغيير عرض رؤوس الصفوف RowHeadersWidthSizeMode:**

توضح كيف يتم ضبط عرض رءوس الصفوف، وهي تأخذ إحدى قيم المرقم DataGridViewRowHeadersWidthSizeMode التالية:

| | |
|---|---|
| EnableResizing | يمكن للمستخدم تغيير عرض رؤوس الصفوف بسحبها بالفأرة. |
| DisableResizing | لا يستطيع المستخدم تغيير عرض رؤوس الصفوف. |
| AutoSizeTo AllHeaders | ضبط عرض رؤوس الصفوف تلقائيا لتناسب محتوياتها. |
| AutoSizeTo DisplayedHeaders | ضبط عرض رؤوس الصفوف الظاهرة على الشاشة تلقائيا لتناسب محتوياتها. |
| AutoSizeTo FirstHeader | ضبط عرض عمود رؤوس الصفوف ليناسب محتوى أول خانة فيه. |

**طراز حافة رؤوس الصفوف RowHeadersBorderStyle:**

تتحكم في شكل إطار رؤوس الصفوف، وهي تأخذ إحدى قيم المرقم DataGridViewHeaderBorderStyle الذي تعرفنا عليه من قبل.

**الطراز الافتراضي لخانات رؤوس الصفوف RowHeadersDefaultCellStyle:**

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في شكل رؤوس الصفوف.

**الطراز المتقدم لحواف رؤوس الصفوف AdvancedRowHeadersBorderStyle:**

تعيد كائن الطراز المتقدم DataGridViewAdvancedBorderStyle الذي يتحكم في شكل إطار رؤوس الصفوف.

**الطراز الافتراضي لخانات الصفوف RowsDefaultCellStyle:**

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في شكل خانات صفوف الجدول.

**الطراز الافتراضي التبادلي لخانات الصفوف AlternatingRowsDefaultCellStyle:**

تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في شكل خانات الصفوف الفردية في الجدول.. لاحظ أنك لو وضعت قيمة في هذه الخاصية، فستتحكم الخاصية RowsDefaultCellStyle في شكل خانات الصفوف الزوجية فقط، بينما تتحكم الخاصية AlternatingRowsDefaultCellStyle في شكل خانات الصفوف الفردية.. لكي ترى تأثير هذا، افتح نافذة الخصائص وحدد هذه الخاصية، واغط زر الانتقال الموجود في خانة قيمتها، لعرض باني الطراز، واستخدمه لجعل لون الخلفية BackColor للصفوف التبادلية فضيا Silver.. سيؤدي هذا إلى أن يعرض الجدول صفا خلفيته بيضاء يليه صف خلفيته فضية ثم صف خلفيته بيضاء وهكذا، كما هو موضح في الصورة:

| About | Phone | CountryID | Author | ID | |
|---|---|---|---|---|---|
| .... | | ٢١ | توفيق الحكيم | ١٢ | ◄ |
| .... | | ٢١ | عباس العقاد | ١٣ | |
| شاعر مصري معاصر | | ٢١ | فاروق جويدة | ١٤ | |
| | | | | | * |

**عرض أخطاء الصفوف ShowRowErrors:**
إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فستظهر أيقونة الخطأ في خانة رأس الصف الذي توجد فيه أخطاء.

**عرض أيقونة التحرير ShowEditingIcon:**
إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فستعرض خانة رأس الصف أيقونة على شكل قلم، عندما يحرر المستخدم قيمة أي خانة في الصف.

**نظام الترتيب SortOrder:**
تحدد اتجاه ترتيب صفوف جدول العرض، وهي تأخذ إحدى قيم المرقم SortOrder التي تعرفنا عليها سابقا.

كما يمدك جدول العرض بالوسائل التالية للتعامل مع الصفوف:

- **تحجيم الصف تلقائيا AutoResizeRow:**
تضبط ارتفاع الصف الذي ترسل إليها رقمه كمعامل، ليناسب محتويات خاناته.
وتوجد صيغة أخرى لهذه الوسيلة، تستقبل إحدى قيم المرقم DataGridViewAutoSizeRowMode، التي توضح كيف يتم تغيير ارتفاع الصف، وقد تعرفنا على هذا المرقم سابقا.

- **تحجيم الصفوف تلقائيا AutoResizeRows:**
تضبط ارتفاع جميع صفوف جدول العرض، لتناسب محتويات خاناتها.
وتوجد صيغة أخرى لهذه الوسيلة، تستقبل إحدى قيم المرقم DataGridViewAutoSizeRowMode، التي توضح كيف يتم تغيير ارتفاع كل صف، وقد تعرفنا على هذا المرقم سابقا.

- **تحجيم عرض رؤوس الصفوف تلقائيا AutoResizeRowHeadersWidth:**
تضبط عرض العمود الذي يحتوي على رؤوس الصفوف، وهي تستقبل إحدى قيم المرقم DataGridViewRowHeadersWidthSizeMode، التي توضح كيف يتم تغيير عرض رؤوس الأعمدة، وقد تعرفنا عليه سابقا.
وتوجد صيغة أخرى لهذه الوسيلة، تزيد على الصيغة السابقة بمعامل أول، يستقبل رقم الصف الذي تريد ضبط العرض تبعا لمحتويات خانته الرئيسية.

**عدد الصفوف المعروضة DisplayedRowCount:**

تعيد عدد الصفوف التي يراها المستخدم على الشاشة في هذه اللحظة، ولها معامل منطقي، إذا جعلت قيمته True فسيدخل ضمن العدد الصفوف التي يظهر جزء منها فقط.

**معرفة مستطيل عرض الصف GetRowDisplayRectangle:**

تعيد كائن مستطيل Rectangle يحتوي على موضع وأبعاد الصف المطلوب، وهي تستقبل معاملين:

- رقم الصف المطلوب.
- معامل منطقي إذا جعلت قيمته True، فستعيد هذه الوسيلة المستطيل المحيط بالجزء المعروض من الصف على الشاشة، وإذا جعلته False، فستعيد المستطيل المحيط بكامل الصف حتى لو كان جزء منه غير ظاهر على الشاشة.

**إبطال الصف InvalidateRow:**

أرسل إليها رقم الصف، لتقوم بإبطال رسمه في الجدول، مما يجبره على إعادة رسم نفسه من جديد.

كما يمدك جدول العرض بالأحداث التالية للتعامل مع الصفوف، علما بأن المعامل الثاني e في معظم هذه الأحداث من النوع DataGridViewRowEventArgs، وهو يمتلك الخاصية Row التي تعيد كائن العمود DataGridViewRow الذي سبب انطلاق الحدث:

**تغير نص خطأ الصف RowErrorTextChanged:**

ينطلق عندما تتغير قيمة الخاصية ErrorText في أحد صفوف الجدول.

**تغير عرض رؤوس الصفوف RowHeadersWidthChanged:**

ينطلق عند تغيير عرض العمود الذي يحتوي على رؤوس الصفوف، سواء بواسطة المستخدم أو من الكود.

**تغير ارتفاع الصف RowHeightChanged:**

ينطلق عندما تتغير قيمة الخاصية Height الخاصة بأحد صفوف الجدول، سواء بواسطة المستخدم، أو من الكود.

## تغير حالة الصف RowStateChanged:

ينطلق عند تغير حالة الصف، مثلما يحدث عند استقباله المؤشر الضوئي Focus أو فقدانه له.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowStateChangedEventArgs، وله الخاصيتان التاليتان:

| | | |
|---|---|---|
| | Row | تعيد كائن الصف DataGridViewRow الذي تغيرت حالته. |
| | StateChanged | تخبرك بحالة الصف التي تغيرت، وهي تعيد إحدى قيم المرقم DataGridViewElementStates التي تعرفنا عليها من قبل. |

على سبيل المثال، لو كان الصف الثاني محددا وغادرته لتحدد الصف الأول، فإن الحدث RowStateChanged سينطلق مرتين كالتالي:

١- المرة الأولى بسبب تغير حالة تحديد الصف الثاني، وستشير الخاصية e.Row إلى الصف الثاني، وستكون للخاصية e.StateChanged القيمة Selected.

٢- المرة الثانية بسبب تغير حالة تحديد الصف الأول، وستشير الخاصية e.Row إلى الصف الأول، وستكون للخاصية e.StateChanged القيمة Selected.

## ضغط رأس الصف RowHeaderMouseClick:

ينطلق عندما يضغط المستخدم رأس الصف بالفأرة.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellMouseEventArgs الذي تعرفنا عليه سابقا.. وقد استخدمنا هذا الحدث في المشروع لعرض القائمة الموضعية المناسبة عند ضغط المستخدم لرأس الصف.. لاحظ أننا لا نستطيع استخدام قوائم موضعية ثابتة في وقت التصميم، لأننا نعرضها فقط إذا حدث خطأ في حفظ الصف الحالي في قاعدة البيانات، كما أن نوع القائمة يختلف تبعا لنوع الخطأ.. وقد استخدمنا المعامل e.RowIndex للحصول على كائن الصف من مجموعة الصفوف:

```
Dim R = DgAuthors.Rows(e.RowIndex)
```

ومن ثم استخدمنا الوسيلة GetRowDisplayRectangle الخاصة بجدول العرض لمعرفة موضع هذا الصف، وذلك لاستخدامه في تحديد موضع القائمة الموضعية:

```
Dim Pos = DgAuthors.GetRowDisplayRectangle(
      e.RowIndex, False).Location
```

بعد هذا فحصنا نص الخطأ الخاص بالصف.. ونظرا لأن نص الخطأ الذي نضعه في الخاصية ErrorText الخاصة بصف مجموعة البيانات ينتقل كما هو إلى الخاصية ErrorText الخاصة بصف جدول العرض، فقد فحصنا بعض الكلمات التي كتبناها لشرح الخطأ، لنعرف منها نوع هذا الخطأ:

```
If R.ErrorText.Contains("حذفه") Then
   InsertCntxt.Show(DgAuthors, Pos + e.Location)
ElseIf R.ErrorText.Contains("بتعديله") Then
     UpdateCntxt.Show(DgAuthors, Pos + e.Location)
End If
```

لاحظ أن موضع مؤشر الفأرة الذي تعيد الخاصية e.Location يكون منسوبا إلى نقطة رأس الصف.. لهذا علينا أن نجمع عليه موضع رأسي الصف ليكون منسوبا إلى النموذج ككل.

**النقر المزدوج على راس الصف RowHeaderMouseDoubleClick:**

ينطلق عندما ينقر المستخدم رأس الصف مرتين بالفأرة.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellMouseEventArgs.

**النقر المزدوج على فاصل الصف RowDividerDoubleClick:**

ينطلق عندما ينقر المستخدم مرتين بالفأرة فوق الخط الفاصل بين صفين، لتحجيم ارتفاع الصف تلقائيا ليناسب محتويات خاناته.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowDividerDoubleClickEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | RowIndex | تعيد رقم الصف الذي نقره المستخدم بالفأرة. |
| | Button | تعيد إحدى قيم المرقم MouseButtons التي تخبرك بزر الفأرة الذي ضغطه المستخدم. |
| | Clicks | تعيد عدد مرات ضغط زر الفأرة. |
| | Delta | تعيد عدد حركات عجلة الفأرة. |
| | X | تعيد الموضع الأفقي لمؤشر الفأرة. |
| | Y | تعيد الموضع الرأسي لمؤشر الفأرة. |
| | Location | تعيد كائن نقطة Point، به موضع مؤشر الفأرة. |
| | Handled | إذا جعلت قيمتها True، فلن تتخذ أية خطوات إضافية لمعالجة الحدث.. هذا معناه إلغاء عملية التحجيم التلقائي للصف. |

## قبل رسم الصف RowPrePaint:

ينطلق قبل رسم أحد صفوف جدول العرض، ليتيح لك التدخل في طريقة رسمه.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowPrePaintEventArgs وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| تعيد رقم الصف. | RowIndex | |
| تعيد كائن المستطيل Rectangle الذي يحمل موضع وأبعاد الصف. | RowBounds | |
| تقرأ أو تغير كائن المستطيل Rectangle الذي يحمل موضع وأبعاد المساحة التي يجب إعادة رسمها من جدول العرض.. لاحظ أن هذه المساحة قد تختلف عن مساحة الصف، فمثلا قد تخفي نافذة أخرى جزءا من جدول العرض، وعند اختفاء هذه النافذة يحتاج الجزء الذي غطته إلى إعادة رسمه لإنعاشه. | ClipBounds | |
| تعيد نص الخطأ الخاص بالصف. | ErrorText | |
| تعيد كائن الرسوم Graphics الذي سيستخدم لرسم الصف. | Graphics | |
| تعيد طراز الخانة DataGridViewCellStyle الذي يتحكم في شكل خانات الصف.. وهي تماثل الخاصية InheritedStyle الخاصة بكائن الصف، لكن استخدام كائن الصف في حدث الرسم يؤدي إلى بطء تنفيذ البرنامج، لذا يفضل استخدام الخاصية InheritedRowStyle لضمان أحسن أداء. | Inherited RowStyle | |
| تعيد True إذا كان الصف المراد رسمه هو أول صف ظاهر على الشاشة. | IsFirst DisplayedRow | |
| تعيد True إذا كان الصف المراد رسمه هو آخر صف مرئي في جدول العرض. | IsLast VisibleRow | |
| تعيد إحدى قيم المرقم DataGridViewElementStates التي توضح حالة الصف المراد رسمه. | State | |

| | | |
|---|---|---|
| | PaintParts | تحدد الأجزاء التي يجب على جدول العرض رسمها في كل خانة من خانات الصف، وهي تأخذ إحدى قيم المرقم DataGridViewPaintParts التالية:<br>- None: لا يتم رسم الخانة.<br>- All: رسم كل أجزاء الخانة.. هذه هي القيمة الافتراضية.<br>- Background: رسم خلفية الخانة.<br>- Border: رسم إطار الخانة.<br>- ContentBackground: رسم خلفية محتوى الخانة.<br>- ContentForeground: رسم لون محتوى الخانة.<br>- ErrorIcon: رسم أيقونة الخطأ.<br>- Focus: رسم المستطيل الذي يشر إلى أن الخانة بها العلامة الضوئية Focus.<br>- SelectionBackground: رسم خلفية التحديد.<br>وتستطيع دمج أكثر من قيمة من هذه القيم معا باستخدام المعامل OR. |
| | Handled | جعل قيمة هذه الخاصية True يخبر جدول العرض بأن حدث الرسم قد تمت الاستجابة له كليا، ولن ينطلق الحدث RowPostPaint ولا الحدث CellPainting.. وعليك ألا تستخدم هذه القيمة إلا إذا أردت إلغاء رسم الصف بواسطة جدول العرض، وفي هذه الحالة عليك أن ترسمه أنت بنفسك من داخل هذا الحدث. |

كما يمتلك المعامل e الوسائل التالية، التي تتيح لك التحكم في رسم مكونات الصف بنفسك:

| | | |
|---|---|---|
| ترسم مستطيلا حول المنطقة التي تريدها، وهي تستقبل المعاملين التاليين:<br>- كائن المستطيل Rectangle الذي سيتم رسم الإطار حوله.<br>- معامل منطقي، إذا جعلته True يتم تلوين المستطيل بلون خلفية التحديد SelectionBackColor، وإذا جعلته False يتم تلوين المستطيل بلون خلفية الصف BackColor.. وكلا اللونين يحددهما طراز الصف DataGridViewRow.InheritedStyle. | DrawFocus | |
| ترسم خانات الصف، وهي تستقبل معاملين:<br>- كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد المساحة التي يراد رسم خاناتها.<br>- إحدى قيم المرقم DataGridViewPaintParts التي توضح الأجزاء المراد رسمها من الخانات. | PaintCells | |
| ترسم خلفية خانات الصف، وهي تستقبل معاملين:<br>- كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد المساحة التي يراد تلوين خاناتها.<br>- معامل منطقي، إذا جعلته True يتم تلوين المستطيل بلون خلفية التحديد SelectionBackColor، وإذا جعلته False يتم تلوين المستطيل بلون الخلفية BackColor. | PaintCells Background | |
| ترسم محتويات خانات الصف، وهي تستقبل كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد المساحة التي يراد رسم محتويات خاناتها. | PaintCells Content | |
| ترسم خانة رأس الصف، ولها صيغتان:<br>١- الأولى تستقبل معاملا منطقيا، إذا جعلته True يتم تلوين خلفية رأس العمود بلون خلفية التحديد، وإذا جعلته False يتم تلوينه بلون الخلفية.<br>٢- الصيغة الثانية تستقبل إحدى قيم المرقم DataGridViewPaintParts التي توضح الأجزاء المراد رسمها من خانة رأس الصف. | PaintHeader | |

### بعد رسم الصف RowPostPaint:

ينطلق بعد رسم أحد صفوف جدول العرض.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowPostPaintEventArgs وهو يمتلك نفس الوسائل والخصائص كما في الحدث السابق ما عدا الخاصيتين PaintParts و Handled.

### إضافة صفوف RowsAdded:

ينطلق عند إضافة صفوف جديدة إلى جدول العرض برمجيا، أو بواسطة المستخدم (عندما يكتب في الصف الجديد الموجود في نهاية الجدول).

ويمكنك استخدام هذا الحدث لترتيب الصفوف المضافة، وذلك باستدعاء الوسيلة Sort الخاصة بجدول العرض.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowsAddedEventArgs، وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| | RowIndex | تعيد رقم أول صف من الصفوف التي أضيفت إلى الجدول. |
| | RowCount | تعيد عدد الصفوف التي أضيفت إلى الجدول. |

### إضافة صف بواسطة المستخدم UserAddedRow:

ينطلق عندما يضيف المستخدم صفا جديدا إلى جدول العرض.

### حذف صفوف RowsRemoved:

ينطلق عند حذف صف أو مجموعة صفوف من جدول العرض، والمعامل الثاني e الخاص به من النوع DataGridViewRowsRemovedEventArgs وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| | RowIndex | تعيد رقم أول صف من الصفوف التي حذفت من الجدول. |
| | RowCount | تعيد عدد الصفوف التي حذفت من الجدول. |

### المستخدم يحذف صفا UserDeletingRow:

ينطلق عندما يحاول المستخدم حذف صف، وقبل أن يتم حذف الصف فعليا، ليتيح لك عرض رسالة تحذير للمستخدم أو إلغاء عملية الحذف.. والمعامل الثاني e لهذا الحدث من نوع الفئة DataGridViewRowCancelEventArgs،

وهي ترث الفئة CancelEventArgs، مما يعني أنك تستطيع وضع القيمة True في الخاصية e.Cancel لإلغاء حذف الصف.. كما يملك هذا المعامل الخاصية e.Row التي تعيد كائن الصف DataGridViewRow الذي يريد المستخدم حذفه.. وقد فعلنا هذا في المشروع DataGridViewAuthorBooks بالكود التالي:

```
If MsgBox("هل تريد حذف هذا الصف",
  MsgBoxStyle.OkCancel) = MsgBoxResult.Cancel Then
        e.Cancel = True
End If
```

كما استخدمنا هذا الحدث في المشروع DataSetSample.. في هذا المشروع لو حذف المستخدم أحد المؤلفين من الجدول العلوي، فإن كتب هذا المؤلف ستظل في جدول الكتب، ولو حاول حفظ التغييرات في قاعدة البيانات فسيتم رفضها بسبب قيد المفتاح الفرعي Foreign Key Constraint المفروض على العلاقة بين المؤلفين وكتبهم، والتي تمنع وجود كتاب مرتبط بمؤلف تم حذفه!

ويمكنك حل هذه المشكلة على مستوى قاعدة البيانات أو مجموعة البيانات، بضبط خصائص القيد لحذف السجلات الفرعية تتابعيا بمجرد حذف السجل الأصلي Cascade Delete.. لكننا استخدمنا حلا آخر في هذا المشروع، وذلك باستخدام الحدث UserDeletingRow الخاص بجدول عرض المؤلفين، لحذف السجلات المعروضة حاليا في جدول عرض الكتب.

### المستخدم حذف صفا UserDeletedRow:

ينطلق بعد حذف المستخدم لأحد صفوف جدول العرض.. لاحظ أن الحدث RowEnter ينطلق أولا قبل انطلاق هذا الحدث، بسبب انتقال المؤشر إلى صف آخر بعد حذف الصف الحالي.. هذا قد يسبب لك مشكلة في بعض المواقف.. مثلا: لا يمكنك استخدام هذا الحدث بدلا من الحدث UserDeletingRow في المشروع DataSetSample، وذلك لأن حذف المؤلف يجعل المؤشر ينتقل إلى مؤلف آخر فينطلق الحدث RowEnter، الذي يعرض كتب المؤلف الحالي في جدول عرض الكتب، ثم ينطلق الحدث UserDeletedRow فيحذف كتب مؤلف موجود، ولا يحذف كتب المؤلف المحذوف!.. لهذا تذكر دائما أن ترتيب انطلاق هذه الأحداث هو:

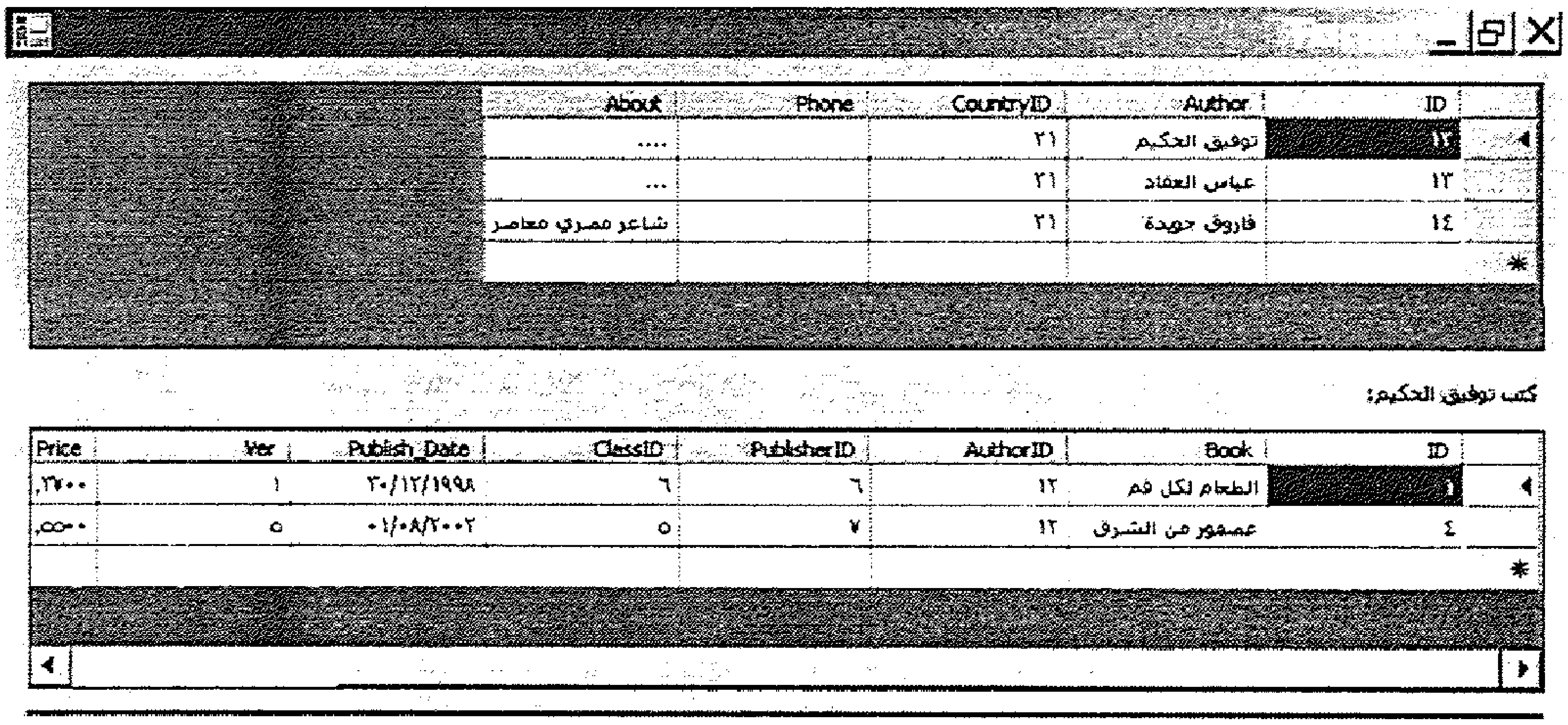
١- UserDeletingRow. ٢- RowEnter. ٣- UserDeletedRow.

### دخول الصف RowEnter:

ينطلق عند استقبال أحد صفوف جدول العرض للمؤشر Focus، لكن قبل أن يصير هو الصف الحالي Current Row.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellEventArgs، وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| | RowIndex | تعيد رقم الصف الذي استقبل المؤشر. |
| | ColumnIndex | تعيد رقم العمود الذي استقبل المؤشر. |

وقد استخدمنا هذا الحدث في المشروع DataGridViewMasterDetails.. في هذا المشروع وضعنا على النموذج جدولي عرض، أحدهما يعرض جدول المؤلفين، والثاني يعرض سجلات الكتب التابعة للمؤلف المحدد حاليا في جدول العرض الأول، كما في الصورة:




ولكي نفعل هذا، اتبعنا الخطوات التالية:

١- استخدمنا الحدث RowEnter لمعرفة الصف الذي تم تحديده:

**Dim R = DGAuthors.Rows(e.RowIndex)**

٢- استخدمنا الخاصية DataGridViewRow.DataBoundItem لنحصل على كائن عرض الصف DataRowView المناظر له في جدول المؤلفين:

**Dim DRv = CType(R.DataBoundItem, DataRowView)**
**If DRv Is Nothing Then Exit Sub**

٣- استخدمنا الوسيلة DataRowView.CreateChildView لإنشاء كائن عرض DataView يحتوي على سجلات الكتب التابعة لسجل هذا المؤلف.. لاحظ أن اسم العلاقة في هذا المشروع بالاسم "كتب المؤلف":

**Dim Dv = DRv.CreateChildView("كتب المؤلف")**

٤- استخدمنا كائن العرض DataView كمصدر بيانات لجدول العرض الثاني، وبهذا يعرض كتب المؤلف المحدد في جدول العرض الأول:

```
DGAuthorBooks.DataSource = Dv
```

وستجد هذا الكود كاملا في المشروع DataGridViewMasterDetails.

وهناك مشروع آخر يعرض المؤلفين وكتبهم بنفس الطريقة، وهو المشروع DataSetSample، لكن الكود الموجود في الحدث RowEnter في هذا المشروع مختلف قليلا، فهو يعتمد على أن رقم الصف في جدول العرض، هو نفس رقم الصف في كائن العرض الافتراضي DefaultView لجدول المؤلفين، لهذا يحصل على كائن عرض الصف كالتالي:

```
Dim TblAuthors = Ds.Tables("Authors")
Dim DRv = TblAuthors.DefaultView(e.RowIndex)
```

وبعد هذا لا يوجد اختلاف في الكود، فهو يحصل على كائن عرض كتب المؤلف، ويستخدمه كمصدر بيانات لجدول عرض الكتب:

```
Dim Dv = DRv.CreateChildView("AuthorsBooks")
DgBooks.DataSource = Dv
```

لاحظ أن هذا الكود سيعمل بشكل صحيح، حتى لو ضغط المستخدم رأي أي عمود لإعادة ترتيب الصفوف رغم أننا نعرف أن تغيير الترتيب يغير رقم كل صف في مجموعة الصفوف.. السبب في عدم حدوث أية مشاكل، أن ترتيب جدول العرض يؤدي إلى ترتيب كائن العرض الافتراضي DefaultView الخاص بجدول المؤلفين، مما يحافظ على نفس ترقيم الصفوف في كل من جدول العرض وكائن العرض، وبالتالي يعمل الكود بشكل صحيح دائما.

### مغادرة الصف RowLeave:

ينطلق عندما يفقد أحد صفوف العرض المؤشر.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellEventArgs كما في الحدث السابق.

### إجازة الصف RowValidating:

ينطلق عندما يحاول المستخدم مغادرة الصف الحالي في جدول العرض.. هذا يتيح لك فحص القيم التي أدخلها في هذا الصف والتأكد من صحتها.

والمعامل الثاني e لهذا الحدث من نوع الفئة DataGridViewCellCancelEventArgs، وهي ترث الفئة

CancelEventArgs، مما يعني أنك تستطيع وضع القيمة True في الخاصية e.Cancel لإجبار المؤشر على البقاء في الصف الحالي، وذلك عندما تكتشف أن به قيمة خاطئة وتريد إجبار المستخدم على تصحيحها أولا.. كما يملك هذا المعامل الخاصيتين e.RowIndex و e.ColumnIndex كما في الحدث RowEnter.

**تمت إجازة الصف RowValidated:**

ينطلق بعد إجازة بيانات الصف الحالي في جدول العرض.. لاحظ أن مغادرة المستخدم للصف الحالي يؤدي إلى انطلاق الأحداث التالية بالترتيب:

١- الحدث RowEnter الخاص بالصف الذي فقد المؤشر.. في الحقيقة هذا أمر عجيب وغير مبرر، لكنه يحدث!
٢- الحدث RowLeave الخاص بالصف الذي فقد المؤشر.
٣- الحدث Validating الخاص بالصف الذي فقد المؤشر.
٤- الحدث Validated الخاص بالصف الذي فقد المؤشر.
٥- الحدث RowEnter الخاص بالصف الجديد الذي استقبل المؤشر.

**إلغاء مشاركة الصف RowUnshared:**

ينطلق عندما يتحول صف مشترك Shared Row إلى صف غير مشترك Unshared Row.. ويمكنك استخدام هذا الحدث أثناء اختبار أداء برنامجك، لمعرفة الكود الذي يتسبب في جعل الصفوف تفقد خاصية المشاركة.. هذا مفيد عندما تريد توفير الذاكرة في البرامج التي تعرض كما هائلا من البيانات في جدول العرض.

## التعامل مع خانات جدول عرض البيانات:

يمنحك جدول العرض الخصائص التالية للتعامل مع خاناته:

### العنصر Item:

هذه هي الخاصية الافتراضية لجدول العرض، وهي تعيد كائن الخانة DataGridViewCell الموجودة في العمود المرسل كمعامل أول، والصف المرسل كمعامل ثان.. والمثال التالي يعرض قيمة الخانة الموجودة في العمود الثاني والصف الثالث:

**MsgBox(DataGridView1(1, 2).Value)**

كما توجد صيغة أخرى، يمكنك أن ترسل إلى معاملها الأول اسم العمود بدلا من رقمه، مثل:

**MsgBox(DataGridView1("Author", 2).Value)**

### الخانة الحالية CurrentCell:

تعيد كائن الخانة DataGridViewCell المحددة حاليا في جدول العرض.. ويمكنك أيضا أن تضع في هذه الخاصية كائن الخانة التي تريد تحديدها، حيث سينزلق جدول العرض لجعلها ظاهرة على الشاشة.

لاحظ أنك لا تستطيع أن تضع في هذه الخاصية خانة رأس الصف Header أو خانة معطلة Disabled أو خانة موجودة في صف مختفي Hidden، وإلا حدث خطأ.

وتعيد هذه الخاصية Nothing إن لم تكن هناك خانة محددة حاليا، ولو وضعت فيها Nothing فسيزول مربع التحديد من الخانة الحالية.

### عنوان الخانة الحالية CurrentCellAddress:

تعيد كائن نقطة Point يحتوي على موضع الخانة الحالية، حيث تمثل الخاصية X رقم الصف الذي توجد به الخانة، والخاصية Y رقم العمود:

**MsgBox(Dgv.CurrentCellAddress.X) ' رقم الصف الحالي**

**MsgBox(Dgv.CurrentCellAddress.Y) ' رقم العمود الحالي**

### هل الخانة الحالية في وضع التحرير IsCurrentCellInEditMode:

تعيد True إذا كانت الخانة الحالية في وضع التحرير حاليا.

### طريقة التحرير EditMode:

تحدد كيف يمكن للمستخدم بدء تحرير الخانة، وهي تأخذ إحدى قيم المرقم DataGridViewEditMode التالية:

| | |
|---|---|
| EditOnEnter | تحرير الخانة بمجرد دخولها.. هذا مفيد عند التنقل بين خانات الصف بضغط زر الجدولة TAB، أو التنقل بين خانات العمود بضغط الزر Enter. |
| EditOnKeystroke | تحرير الخانة عند ضغط أي حرف أبجدي. |
| EditOn KeystrokeOrF2 | تحرير الخانة عند ضغط حرف أبجدي أو الزر F2.. هذه هي القيمة الافتراضية. |
| EditOnF2 | تحرير الخانة عند ضغط الزر F2. |
| Edit Programmatically | تحرير الخانة برمجيا باستدعاء الوسيلة BeginEdit. |

لاحظ أن كل القيم السابقة ما عدا EditProgrammatically تسمح للمستخدم ببدء تحرير الخانة بمجرد نقرها مرتين بالفأرة.

### أداة التحرير EditingControl:

تعيد كائن الأداة Control الذي يحمل أداة التحرير المستضافة في الخانة الحالية، إن كانت في وضع التحرير.. وإذا لم تكن الخانة الحالية في وضع التحري، تعيد هذه الخاصية Nothing.

### لوحة التحرير EditingPanel:

تعيد كائن اللوحة Panel التي تستخدم لعرض أداة التحرير في الخانة الحالية.. ولهذه الخاصية قيمة دائما حتى لو لم تكن الخانة في وضع التحرير.

### هل الخانة الحالية قذرة IsCurrentCellDirty:

تعيد True إذا كانت قيمة الخانة الحالية قد تغيرت ولم يتم حفظ التغيير في مصدر البيانات بعد.

### أول خانة معروضة FirstDisplayedCell:

تعيد كائن الخانة DataGridViewCell الذي يمثل أول خانة عادية (ليست رأس صف أو عمود) ظاهرة على الشاشة في جدول العرض.. هذه الخانة تكون أعلى يسار الجدول المعروض من اليسار إلى اليمين، وأعلى يمين الجدول المعروض من اليمين إلى اليسار.. ويمكنك أيضا أن تضع في هذه الخاصية كائن الخانة التي تريد أن ينزلق جدول العرض إليها ليجعلها أول خانة معروضة فيه.

### الخانة العلوية اليسرى TopLeftHeaderCell:

تقرأ أو تغير كائن الخانة الرئيسية DataGridViewHeaderCell الذي يمثل الخانة الموجودة في الركن العلوي الأيسر من جدول العرض المعروض من اليسار إلى اليمين، أو الموجودة في الركن العلوي الأيمن في الجدول المعروض من اليمين إلى اليسار.. ويمكنك الاستفادة من هذه الخاصية في التحكم في خصائص هذه الخانة، ككتابة نص بها، أو وضع قامة موضعية لها... إلخ.

### الخانات المحددة SelectedCells:

تعيد مجموعة من النوع DataGridViewSelectedCellCollection، التي ترث الفئة BaseCollection وتمثل الواجهة IList، وهي تحتوي على الخانات المحددة حاليا في جدول العرض.

### عرض أخطاء الخانات ShowCellErrors:

إذا جعلت قيمة هذه الخاصية True، فستعرض الخانات التي بها أخطاء أيقونة حمراء لتنبيه المستخدم.

### عرض تلميحات الخانات ShowCellToolTips:

إذا جعلت قيمة هذه الخاصية True، فسيظهر تلميح على الشاشة عندما يحلق المستخدم بالفأرة فوق أي خانة من خانات الجدول.. لاحظ أن هذا التلميح هو النص الموجود في الخاصية ToolTipText الخاصة بالخانة.

### طريقة النسخ إلى لوحة القصاصات ClipboardCopyMode:

تتحكم في كيفية نسخ الخانات المحددة إلى لوحة القصاصات، وهي تأخذ إحدى قيم المرقم DataGridViewClipboardCopyMode التالية:

| | |
|---|---|
| Disable | لا يسمح بنسخ محتويات الخانات إلى لوحة القصاصات. |
| EnableWith AutoHeaderText | يتم نسخ محتويات الخانات المحددة، وإن كانت هناك رؤوس أعمدة أو رؤوس صفوف محددة يتم نسخ محتوياتها أيضا. |
| EnableWithout HeaderText | يتم نسخ محتويات الخانات المحددة فقط، مع تجاهل رؤوس الصفوف والأعمدة. |
| EnableAlways IncludeHeaderText | يتم نسخ محتويات الخانات المحددة، ونسخ رؤوس الأعمدة ورؤوس الصفوف التي توجد فيها هذه الخانات، بغض النظر عما إذا كانت هذه الرؤوس محددة أم لا. |

### طراز حواف الخانات CellBorderStyle:

تحدد شكل إطار خانات الجدول، وهي تأخذ إحدى قيم المرقم DataGridViewCellBorderStyle التالية:

| | |
|---|---|
| None | لا توجد إطارات. |
| Single | إطار يتكون من خط مفرد. |
| SingleVertical | حافة رأسية تتكون من خط مفرد. |
| SingleHorizontal | حافة أفقية تتكون من خط مفرد. |
| Sunken | إطار غائر. |
| SunkenVertical | حافة رأسية غائرة. |
| SunkenHorizontal | حافة أفقية غائرة. |
| Raised | إطار بارز. |
| RaisedVertical | حافة رأسية بارزة. |
| RaisedHorizontal | حافة أفقية بارزة. |
| Custom | إطار مخصص.. هذه القيمة للقراءة فقط، ولا يمكنك وضعها بنفسك، وإنما تتغير تلقائيا عند تغيير الخاصية AdvancedCellBorderStyle. |

### الطراز المتقدم لحواف الخانات AdvancedCellBorderStyle:

تعيد كائن الطراز المتقدم DataGridViewAdvancedBorderStyle الذي يتحكم في شكل إطار خانات جدول العرض.

**الطراز الافتراضي للخانات DefaultCellStyle:**
تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في مظهر خانات الجدول.

كما يمدك جدول العرض بالوسائل التالية للتعامل مع الخانات:

**تحديد الكل SelectAll:**
تحدد كل خانات جدول العرض.

**هل جميع الخانات محددة AreAllCellsSelected:**
تعيد True إذا كانت جميع خانات جدول العرض محددة، وهي تستقبل معاملا منطقيا، إذا جعلته False فسيتم فحص الخانات المرئية Visible فقط، أما إذا جعلته True فستدخل الخانات الخفية في الاعتبار.

**إزالة التحديد ClearSelection:**
تجعل كل خانات جدول العرض غير محددة.
ولهذه الوسيلة صيغة ثانية، تتيح لك استثناء خانة معينة من هذه العملية، وهي تستقبل المعاملات التالية:
- رقم العمود الذي توجد به الخانة المستثناة.
- رقم الصف الذي توجد به الخانة المستثناة.
- معامل منطقي، إذا جعلته True فسيتم تحديد الخانة المستثناة إن لم تكن محددة فعلا، وإذا جعلته False فستترك الخانة على حالتها الأصلية كما كانت، سواء كانت محددة أم لا.

وقد استخدمنا هذه الخاصية في الزر "تحديد الصف الثاني" في المشروع DataGridViewAuthorBooks لإزالة تحديد كل الخانات قبل تحديد الصف الثاني كالتالي:

```
DGAuthors.ClearSelection( )
DGAuthors.Rows(0).Selected = True
```

لاحظ أن هذا الكود أكفأ من كود الزر "تحديد الصف الأول" في نفس المشروع، ليس فقط لأنه أكثر اختصارا وسهولة، ولكن لأن إزالة تحديد الصفوف المحددة لا يزيل تحديد الخانات المتفرقة المحددة.. جرب أن تضغط الزر CTRL وتضغط أكثر من خانة متفرقة بالفأرة.. لو ضغطت الزر "تحديد الصف الأول" فسيحدد الصف الأول دون إزالة تحديد هذه الخانات، بينما لو ضغطت الزر الثاني، فسيزيل تحديد جميع هذه الخانات ويحدد الصف الثاني.

### الحصول على محتوى لوحة القصاصات GetClipboardContent:

تعيد كائن بيانات DataObject، يحتوي على الخانات المحددة حاليا في جدول العرض، ليمكنك وضعه في لوحة القصاصات مباشرة دون أن تشغل ذهنك بنسخ محتويات الخانات بتنسيق خاص بك.. ولقد شرحنا كائن البيانات DataObject بالتفصيل عندما تعرفنا على لوحة القصاصات Clipboard في كتاب "برمجة نماذج الويندوز".

وتسبب هذه الوسيلة خطأ في البرنامج إذا كانت للخاصية ClipboardCopyMode القيمة Disable.

وما لم تكن في حاجة إلى برمجة أوامر النسخ واللصق الخاصة بك في قائمة رئيسية أو موضعية، فلن تحتاج إلى استدعاء هذه الوسيلة بنفسك، فبمجرد ضغط المستخدم Ctrl+C من لوحة المفاتيح يتم نسخ الخانات المحددة في جدول العرض إلى لوحة القصاصات تلقائيا، حيث يمكنك لصقها في أي برنامج آخر.. على سبيل المثال: عند لصق الخانات في برنامج Notepad يوضع حرف جدولة (أربع مسافات) بين محتوى كل خانة والتي تليها، ويوضع كل صف في سطر جديد.. أما عند لصق هذه الخانات المنسوخة في برنامج Word، فإنه يعرضها في شكل جدول.

### معرفة عدد الخانات GetCellCount:

تعيد عدد خانات جدول العرض التي لها الحالة المرسلة كمعامل، وهي تستقبل إحدى قيم المرقم DataGridViewElementStates التي تعرفنا عليها من قبل.

### معرفة مستطيل عرض الخانة GetCellDisplayRectangle:

تعيد كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد الخانة المطلوبة، وهي تستقبل المعاملات التالية:

- رقم العمود الذي توجد به الخانة.
- رقم الصف الذي توجد به الخانة.
- معامل منطقي إذا جعلته True فسيحتوي المستطيل على أبعاد الجزء الظاهر من الخانة على الشاشة، وإذا جعلته False فسيحتوي المستطيل على أبعاد الخانة كلها.

- **إبطال الخانة InvalidateCell:**

أرسل إلى هذه الوسيلة كائن الخانة DataGridViewCell التي تريد تعطيل رسمها لتجبر جدول العرض على إعادة رسمها من جديد.
وتوجد صيغة أخرى من هذه الوسيلة تستقبل رقم العمود ورقم الصف اللذين توجد بهما الخانة بدلا من استقبال كائن الخانة.

- **بدء التحرير BeginEdit:**

تضع الخانة الحالية في وضع التحرير.. ولهذه الوسيلة معامل منطقي، إذا جعلته True فسيتم تحديد كل محتويات الخانة في أداة التحرير.

- **إلغاء التحرير CancelEdit:**

تلغي تحرير الخانة الحالية وتعيد الخانة إلى قيمته الأصلية.. وتعيد هذه الوسيلة True إذا نجح إلغاء التحرير.

- **قبول التحرير CommitEdit:**

تحفظ القيمة من أداة التحرير إلى الخانة الحالية، دون إنهاء وضع التحرير.. وهي تستقبل كمعامل إحدى قيم المرقم DataGridViewDataErrorContexts التي توضح الخطأ الذي يمكن أن يحدث أثناء حفظ القيمة، وقد تعرفنا على هذا المرقم من قبل.. وتعيد هذه الوسيلة True إذا نجحت عملية الحفظ.

- **إنعاش التحرير RefreshEdit:**

تحدث القيمة التي تعرضها أداة التحرير، بإعادة قراءة القيمة من الخانة الحالية.. هذا مفيد عندما تتغير قيمة الخانة الحالية (نتيجة تغير مصدر البيانات) بينما يقوم المستخدم بتحرير الخانة، وتريد أنت تنبيهه إلى هذا التغيير.. وتعيد هذه الوسيلة True إذا نجح إنعاش أداة التحرير.

- **إنهاء التحرير EndEdit:**

تنهي تحرير الخانة الحالية وتحفظ القيمة الجديدة في الخانة الحالية، وتعيد True إذا نجح إنهاء التحرير.
وتوجد صيغة أخرى لهذه الوسيلة، تستقبل كمعامل إحدى قيم المرقم DataGridViewDataErrorContexts التي توضح الخطأ الذي يمكن أن يحدث أثناء حفظ القيمة، وقد تعرفنا على هذا المرقم من قبل.

**التنبيه بأن الخانة الحالية قذرة NotifyCurrentCellDirty:**

تنبه جدول العرض إن كانت الخانة الحالية قد حفظت التغييرات إلى مصدر البيانات أم لا.. وهي تستقبل معاملا منطقيا، إذا جعلته True كان هذا معناه أن الخانة الحالية لم تحفظ التغييرات إلى مصدر البيانات.

وعليك استخدام هذه الوسيلة إذا كنت تتعامل مع أنواع خانات خاصة بك.. لهذا استخدمناها في المشروع DataGridColumnTypes في الحدث الدال على تغير محتويات أدوات التحرير الجديدة التي أنشأناها:

- الحدث OnValueChanged في الفئة CalendarEditingControl.
- والحدث OnTextChanged في الفئتين TreeEditingControl و TreeComboEditingControl.

كما يمدك جدول العرض بالأحداث التالية للتعامل مع الخانات.. والمعامل الثاني e لمعظم هذه الأحداث من النوع DataGridViewCellEventHandler الذي تعرفنا عليه سابقا، وهو يخبرك برقم الصف ورقم العمود الذي توجد به الخانة:

**تغير نص خطأ الخانة CellErrorTextChanged:**

ينطلق عندما تتغير قيمة الخاصية ErrorText الخاصة بإحدى الخانات.

**تغير حالة الخانة CellStateChanged:**

ينطلق عندما تتغير حالة إحدى خانات جدول العرض، كأن يوضع بها المؤشر الضوئي أو يزول منها.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellStateChangedEventArgs، وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| | Cell | تعيد كائن الخانة DataGridViewCell التي تغيرت حالتها. |
| | StateChanged | تخبرك بالحالة الجديدة للخانة، وهي تعيد إحدى قيم المرقم DataGridViewElementStates الذي تعرفنا عليه من قبل. |

### تغير الخانة الحالية CurrentCellChanged:

ينطلق عندما تتغير الخانة المحددة حاليا في جدول العرض، بسبب تغير قيمة الخاصية CurrentCell من الكود، أو بسبب انتقال المستخدم من الخانة الحالية إلى خانة أخرى.

### تغير قيمة الخانة CellValueChanged:

ينطلق عندما تتغير قيمة إحدى خانات جدول العرض، وذلك بعد انتهاء وضع التحرير.. ويمكنك استخدام هذا الحدث لاتخاذ رد الفعل المناسب بعد تغير قيمة الخانة، مثل فحص القيمة الجديدة للتأكد من صحتها، أو إعادة ترتيب صفوف الجدول إذا كان الجدول مرتبا تبعا لخانات العمود الذي توجد به هذه الخانة.

### تغير الحالة القذرة للخانة الحالية CurrentCellDirtyStateChanged:

ينطلق عندما تتغير قيمة الخانة بينما لا زالت في وضع التحرير ولم يتم حفظ قيمتها فعلا.. هذا مفيد في بعض الحالات، مثل الاستجابة لتغيير المستخدم لحالة مربع اختيار CheckBox موضوع في إحدى الخانات دون مغادرة الخانة، ففي هذه الحالة لن ينطلق الحدث CellValueChanged.. وقد استخدمنا هذا الحدث في المشروع DataGridColumnTypes لعرض رسالة تخبر المستخدم بحالة مربع الاختيار بمجرد تغييرها، ودون حتى أن يغادر الخانة.. لفعل هذا، فعلنا ما يلي:

- تأكدنا أولا أن الخانة الحالية في جدول العرض CurrentCell هي خانة مربع اختيار DataGridViewCheckBoxCell، لأن هذا الحدث ينطلق مع أي نوع من أنواع الخانات.
- استخدمنا الخاصية EditedFormattedValue الخاصة بالخانة الحالية لعرض قيمة مربع الاختيار للمستخدم.. ونظرا لأن هذه الخاصية تعيد كائنا Object، فقد حولناه أولا إلى نوع المرقم CheckState لنستطيع عرض اسم حالة الاختيار باستخدام الوسيلة ToString، ولو لم نفعل هذا، فستظهر أرقام تدل على الحالة مثل (٠ و ١ و ٢).
- ستواجهنا مشكلة هنا، وهي أن هذا الحدث ينطلق بعد أول مرة تتغير فيها حالة الخانة، ومهما وضع المستخدم علامة الاختيار أو أزالها فلن ينطلق هذا الحدث، إلا إذا غادر المستخدم الخانة الحالية أولا ليتم حفظ

التغيير.. ولحل هذه المشكلة يمكننا أن نجبر الخانة على حفظ التغييرات باستخدام الوسيلة CommitEdit الخاصة بجدول العرض.

- لكن استدعاء الوسيلة CommitEdit سيؤدي إلى انطلاق الحدث CurrentCellDirtyStateChanged في الحال وقبل تنفيذ باقي الإجراء الحالي، وهو ما سيجعل الرسالة تظهر مرتين.. ولحل هذه المشكلة، عرّفنا متغيرا منطقيا Boolean Variable اسمه ExitCurrentCellDirtyStateChanged، واستخدمناه كمؤشر Flag، بحيث نجعل قيمته True قبل استدعاء الوسيلة CommitEdit ونعيدها إلى False بعدها مباشرة، وفي بداية الحدث CurrentCellDirtyStateChanged نفحص قيمة هذا المتغير، فإن كانت True نغادر الإجراء في الحال.. وبهذا تظهر الرسالة مرة واحدة فقط.

وستجد هذا الكود كاملا في المشروع DataGridColumnTypes.

### بدء تحرير الخانة CellBeginEdit:

ينطلق عند بدء تحرير إحدى خانات جدول العرض.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellCancelEventArgs الذي تعرفنا عليه من قبل، وهو يتيح لك إلغاء عملية التحرير بوضع القيمة True في الخاصية e.Cancel.

### ظهور أداة التحرير EditingControlShowing:

ينطلق عند بدء تحرير الخانة الحالية في جدول العرض، وظهور أداة التحرير بها.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewEditingControlShowingEventArgs، وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| 🔒 | Control | تعيد كائن الأداة Control الذي يمثل أداة التحرير التي يتم عرضها حاليا.. ويمكنك تحويل هذا الكائن إلى نوع أداة التحرير الفعلي، واستخدامه لتغيير خصائصها أو وضع القيم الابتدائية بها. |
| 🔒 | CellStyle | تعيد كائن طراز الخانة DataGridViewCellStyle، الذي يمكنك من خلاله التحكم في شكل وتنسيق الخانة التي يتم تحريرها حاليا. |

- **إنهاء تحرير الخانة CellEndEdit:**

ينطلق عند انتهاء تحرير الخانة الحالية في جدول العرض.

- **ضغط الخانة CellClick:**

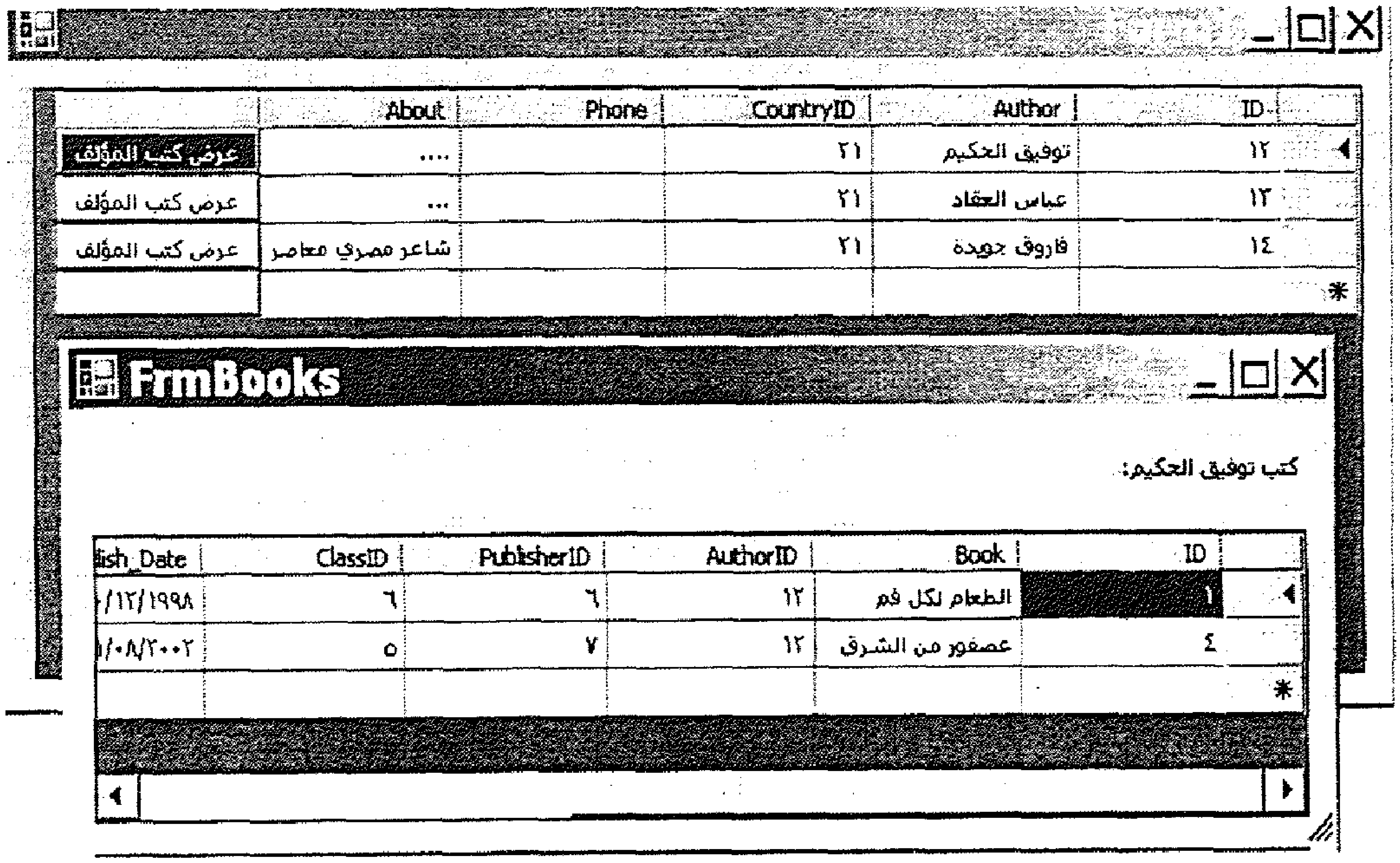
ينطلق عند ضغط أي جزء من الخانة بزر الفأرة بما في ذلك إطارها وهامشها ومحتوياتها وأي أداة موضوعة داخلها.. وينطلق أيضا عند ضغط زر المسافة من لوحة المفاتيح عندما يكون بالخانة زر أو مربع اختيار.

- **ضغط الخانة بالفأرة CellMouseClick:**

مماثل للحدث السابق، ولكنه يتميز عنه بأن المعامل الثاني e من النوع DataGridViewCellMouseEventArgs الذي تعرفنا عليه سابقا، وهو يعطيك معلومات وافية عن موضع الفأرة وحالة أزرارها.

- **ضغط محتويات الخانة CellContentClick:**

ينطلق عند ضغط الأداة التي تستضيفها الخانة.. وقد استخدمنا هذا الحدث في التطبيق DataGridColumnTypes لعرض رسالة للمستخدم عندما يضغط زرا في أحد خانات عمود الأزرار، كما استخدمناه في المشروع DataGridViewAuthorBooks، لعرض نافذة جديدة بها كتب المؤلف الذي تم ضغط الزر الخاص به.



- **النقر المزدوج على الخانة CellDoubleClick:**
ينطلق عند النقر مرتين بالفأرة على أي جزء من الخانة، بما في ذلك إطارها وهامشها ومحتوياتها وأي أداة موضوعة داخلها.

- **النقر المزدوج على الخانة بالفأرة CellMouseDoubleClick:**
مماثل للحدث السابق، ولكنه يتميز عنه بأن المعامل الثاني e من النوع DataGridViewCellMouseEventArgs الذي تعرفنا عليه سابقا، وهو يعطيك معلومات وافية عن موضع الفأرة وحالة أزرارها.

- **النقر المزدوج على محتويات الخانة CellContentDoubleClick:**
ينطلق عند النقر مرتين بالفأرة على الأداة التي تستضيفها الخانة.

- **هبوط زر الفأرة فوق الخانة CellMouseDown:**
يحدث مباشرة بعد ضغط المستخدم لأحد أزرار الفأرة، بينما مؤشرها فوق الخانة، وقبل أن يترك الزر.. والمعامل الثاني e من النوع DataGridViewCellMouseEventArgs.

- **ارتفاع زر الفأرة فوق الخانة CellMouseUp:**
ينطلق بعد ترك المستخدم لزر الفأرة المضغوط، ومؤشرها فوق الخانة.. والمعامل الثاني e من النوع DataGridViewCellMouseEventArgs.

- **دخول الفأرة إلى الخانة CellMouseEnter:**
ينطلق عندما يدخل مؤشر الفأرة إلى حدود الخانة، وينطلق لمرة واحدة فقط إلى أن يغادر المؤشر حدود الخانة.

- **مغادرة الفارة للخانة CellMouseLeave:**
ينطلق عندما يغادر مؤشر الفأرة حدود الخانة.

- **تحرك الفأرة فوق الخانة CellMouseMove:**
ينطلق أثناء تحرك مؤشر الفأرة فوق الخانة.. والمعامل الثاني e من النوع DataGridViewCellMouseEventArgs، وهو ينطلق عدة مرات في كل ثانية إلى أن يتوقف المؤشر عن الحركة، أو يغادر حدود الخانة.

### دخول الخانة CellEnter:
ينطلق عندما تتغير الخانة الحالية في جدول العرض بالانتقال إلى خانة أخرى، أو عندما ينتقل المؤشر الضوئي من أداة أخرى إلى جدول العرض.
ويمكن أن ينطلق هذا الحدث مرتين متتاليتين، وذلك إذا كان المؤشر الضوئي في أداة أخرى، وضغط المستخدم خانة غير الخانة التي كانت محددة في جدول العرض.

### مغادرة الخانة CellLeave:
ينطلق عندما يغادر المستخدم الخانة الحالية في جدول العرض، وتفقد المؤشر الضوئي.

### تجري إجازة الخانة CellValidating:
ينطلق عندما يحاول المستخدم مغادرة الخانة الحالية.. والمعامل الثاني e لهذا الحدث من نوع الفئة DataGridViewCellValidatingEventArgs، وهي ترث الفئة CancelEventArgs، مما يعني أنك تستطيع وضع القيمة True في الخاصية e.Cancel لإجبار المؤشر على البقاء في الخانة الحالية، وذلك عندما تكتشف أن بها قيمة خاطئة وتريد إجبار المستخدم على تصحيحها أولا.. وإضافة إلى هذا، يمتلك هذا المعامل الخصائص التالية:

| | | |
|---|---|---|
| 🔒 | ColumnIndex | تعيد رقم العمود الذي توجد به الخانة. |
| 🔒 | RowIndex | تعيد رقم الصف الذي توجد به الخانة. |
| 🔒 | FormattedValue | تعيد القيمة المنسقة الموجودة في الخانة، والتي عليك التأكد من صحتها. |

### تمت إجازة الخانة CellValidated:
ينطلق بعد التأكد من صحة القيمة الموجودة في الخانة.

### تنسيق الخانة CellFormatting:
ينطلق عندما تحتاج الخانة إلى عرض قيمتها، لتسمح لك بتنسيقها بالشكل الذي تريده.. ويمكنك استخدام الخاصية DataGridViewCellStyle.Format مباشرة لتحديد صيغة تنسيق الخانة، لكن أحيانا قد لا تجد صيغة مباشرة للتنسيق الذي تريده، وفي هذه الحالة يمكنك استخدام هذا الحدث لوضع القيمة

بالشكل الذي تريده في الخانة.. كما أن هذا الحدث يتيح لك تغيير شكل الخانة وليس محتوياتها فقط.
لاحظ أن هذا الحدث ينطلق كلما تم إنعاش رسم الخانة في الجدول، وكلما أردت قراءة قيمتها المنسقة.. هذا معناه أن هذا الحدث ينطلق كثيرا، لهذا عليك ألا تكتب فيه أي كود طويل يستهلك وقتا ملموسا، حتى لا تؤثر على سرعة وأداء البرنامج.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellFormattingEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| 🔒 | ColumnIndex | تعيد رقم العمود الذي توجد به الخانة. |
| | RowIndex | تعيد رقم الصف الذي توجد به الخانة. |
| 🔒 | CellStyle | تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في مظهرها وتنسيقها. |
| 🔒 | DesiredType | تعيد كائن النوع Type، الذي يمثل نوع القيمة المنسقة المراد عرضها في الخانة. |
| | Value | تحتوي القيمة الأصلية للخانة، وعليك أن تضع بدلا منها القيمة المنسقة التي تريد عرضها. |
| 🔒 | Formatting Applied | إذا جعلت قيمة هذه الخاصية True، فستخبر الخانة بأن القيمة الموجودة في الخاصية Value هي القيمة المنسقة، وعليها عرضها مباشرة بدون أي عمليات تنسيق إضافية.. والقيمة الافتراضية لهذه الخاصية هي False، وهذا معناه أن على الخانة استخدام خصائص التنسيق الموجودة في الخاصية CellStyle. |

### تحويل قيمة الخانة CellParsing:

ينطلق بعد انتهاء تحرير قيمة الخانة، وذلك ليسمح لك بتحويل القيمة المنسقة التي أدخلها المستخدم إلى النوع الأصلي لبيانات الخانة.. هذا مفيد عندما تتعامل مع تنسيق خاص بك باستخدام الحدث السابق، وتريد إجراء عملية التحديث العكسية.. ولو لم تستخدم هذا الحدث، فستقوم الخانة بالتحويل التلقائي من النوع المنسق إلى النوع الأصلي.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellParsingEventArgs، وهو يمتلك الخصائص التالية:

| | |
|---|---|
| ColumnIndex | تعيد رقم العمود الذي توجد به الخانة. |
| RowIndex | تعيد رقم الصف الذي توجد به الخانة. |
| Inherited CellStyle | تقرأ أو تغير كائن طراز الخانة DataGridViewCellStyle الذي يتحكم في مظهرها وتنسيقها. |
| DesiredType | تعيد كائن النوع Type، الذي يمثل نوع القيمة الأصلية المراد التحويل إليها. |
| Value | تحتوي القيمة المنسقة للخانة، وعليك أن تضع بدلا منها القيمة التي تريد حفظها في الخانة. |
| Parsing Applied | إذا جعلت قيمة هذه الخاصية True، فستخبر الخانة بأن القيمة الموجودة في الخاصية Value هي القيمة الأصلية، وعليها حفظها مباشرة بدون أي عمليات تحويل إضافية.. والقيمة الافتراضية لهذه الخاصية هي False، وهذا معناه أن على الخانة استخدام خصائص التنسيق الموجودة في الخاصية InheritedCellStyle لتحويل القيمة المنسقة إلى القيمة الأصلية. |

### رسم الخانة CellPainting:

ينطلق عندما يحتاج جدول العرض إلى إعادة رسم إحدى خاناته.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellPaintingEventArgs، وهو يمتلك الخصائص التالية:

| | |
|---|---|
| ColumnIndex | تعيد رقم العمود الذي يحتوي على الخانة. |
| RowIndex | تعيد رقم الصف الذي يحتوي على الخانة. |
| CellStyle | تعيد كائن طراز الخانة DataGridViewCellStyle، الذي يتحكم في شكل وتنسيق الخانة المراد رسمها. |
| Advanced BorderStyle | تعيد كائن الطراز المتقدم DataGridViewAdvancedBorderStyle الذي يتحكم في شكل إطار الخانة. |

| | | |
|---|---|---|
| تعيد كائن المستطيل Rectangle الذي يحمل موضع وأبعاد الخانة المراد رسمها. | CellBounds | |
| تقرأ أو تغير كائن المستطيل Rectangle الذي يحمل موضع وأبعاد المساحة التي يجب إعادة رسمها من الخانة. | ClipBounds | |
| تعيد نص الخطأ الخاص بالخانة. | ErrorText | |
| تعيد كائن الرسوم Graphics الذي سيستخدم لرسم الخانة. | Graphics | |
| تعيد إحدى قيم المرقم DataGridViewElementStates التي توضح حالة الخانة المراد رسمها. | State | |
| تعيد قيمة الخانة. | Value | |
| تعيد القيمة المنسقة للخانة. | Formatted Value | |
| تحدد الأجزاء التي يجب على جدول العرض رسمها في الخانة، وهي تأخذ إحدى قيم المرقم DataGridViewPaintParts. | PaintParts | |
| إذا جعلت قيمة هذه الخاصية True فستخبر جدول العرض بأن حدث الرسم قد تمت الاستجابة له كليا.. وعليك ألا تستخدم هذه القيمة إلا إذا أردت إلغاء رسم جدول العرض للخانة، وفي هذه الحالة عليك أن ترسمها أنت بنفسك من داخل هذا الحدث. | Handled | |

كما يمتلك المعامل e الوسائل التالية:

| | | |
|---|---|---|
| ترسم جزءا من الخانة، وهي تستقبل معاملين:<br>- كائن المستطيل Rectangle الذي يحمل موضع وأبعاد المساحة التي سيعاد رسمها من الخانة.<br>- إحدى قيم المرقم DataGridViewPaintParts تحدد أجزاء الخانة التي يجب رسمها. | Paint | |

| | | |
|---|---|---|
| | Paint Background | ترسم خلفية الخانة، وهي تستقبل معاملين:<br>- كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد المساحة التي يراد رسم خلفيتها.<br>- معامل منطقي، إذا جعلته True فسيتم تلوين المستطيل بلون خلفية التحديد SelectionBackColor، وإذا جعلته False فسيتم تلوين المستطيل بلون الخلفية BackColor. |
| | PaintContent | ترسم محتويات الخانة، وهي تستقبل كائن المستطيل Rectangle الذي يحتوي على موضع وأبعاد المساحة التي يراد رسمها.. لاحظ أن عليك رسم خلفية الخانة أولا قبل رسم محتوياتها، ولو فعلت العكس فسيمحو رسم الخلفية محتويات الخانة! |

**التعامل مع جدول العرض:**

يمنحك جدول العرض الخصائص التالية للتحكم في مظهره وأدائه:

**لون الخلفية BackgroundColor:**

تتحكم في لون خلفية جدول العرض، وهو اللون الظاهر في الجزء الذي لا توجد به خانات.

**طراز الحواف BorderStyle:**

تحدد شكل إطار جدول العرض، وهي تأخذ إحدى قيم المرقم BorderStyle التالية:

| | |
|---|---|
| None | جدول العرض بدون أي إطار. |
| FixedSingle | يحيط بجدول العرض مستطيل أسود يتكون من خط مفرد السماكة. |
| Fixed3D | إطار مجسم (ثلاثي الأبعاد). |

### لون الشبكة GridColor:

تتحكم في لون شبكة الخطوط التي تفصل بين الصفوف والأعمدة.. وتؤثر هذه الخاصية فقط على الإطار المفرد، لكنها تكون بلا تأثير عند استخدام إطارات مجسمة ففي هذه الحالة تستخدم الألوان الخاصة بنظام التشغيل.

### متعددة التحديد MultiSelect:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيستطيع المستخدم تحديد أكثر من خانة أو صف أو عمود معا في نفس الوقت.

### للقراءة فقط ReadOnly:

إذا جعلت قيمة هذه الخاصية True، فلن يستطيع المستخدم تحرير خانات جدول العرض.. والقيمة الافتراضية هي False.

### المنزلقات ScrollBars:

توضح أيا من المنزلقين سيعرضه جدول العرض، وهي تأخذ إحدى قيم المرقم ScrollBars التالية:

| | |
|---|---|
| None | لا تظهر أية منزلقات. |
| Horizontal | المنزلق الأفقي فقط. |
| Vertical | المنزلق الرأسي فقط. |
| Both | المنزلقين الأفقي والرأسي معا.. هذه هي القيمة الافتراضية. |

### إزاحة الانزلاق الأفقي HorizontalScrollingOffset:

تتحكم في الموضع المبدئي للمنزلق الأفقي.. والكود التالي يجعل المنزلق الأفقي يتحرك ليجعل العمود الثاني في الجدول أول عمود ظاهر على الشاشة:

```
Dgv.HorizontalScrollingOffset = Dgv.Columns(0).Width
```

### إزاحة الانزلاق الرأسي VerticalScrollingOffset:

تعيد موضع المنزلق الرأسي لجدول العرض.. وعلى عكس الخاصية السابقة، لا يمكنك تغيير موضع المنزلق الرأسي بنفسك، وهذا أمر عجيب!

### طريقة التحديد SelectionMode:

تتحكم في طريقة تحديد خانات جدول العرض، وهي تأخذ إحدى قيم المرقم DataGridViewSelectionMode التالية:

| | |
|---|---|
| CellSelect | يمكن تحديد خانة أو أكثر. |
| FullRow Select | يؤدي ضغط رأس الصف أو أية خانة موجودة به إلى تحديد الصف كله. |
| FullColumn Select | يؤدي ضغط رأس العمود أو أية خانة موجودة به إلى تحديد العمود كله.. وتسبب هذه القيمة خطأ في البرنامج إذا كانت للخاصية SortMode الخاصة بالعمود القيمة Automatic. |
| RowHeader Select | يؤدي ضغط رأس الصف إلى تحديد الصف كله، بينما يؤدي ضغط أي خانة إلى تحديد هذه الخانة بمفردها.. هذه هي القيمة الافتراضية. |
| ColumnHeader Select | يؤدي ضغط رأس العمود إلى تحديد العمود كله، بينما يؤدي ضغط أي خانة إلى تحديد هذه الخانة بمفردها.. وتسبب هذه القيمة خطأ في البرنامج إذا كانت للخاصية SortMode الخاصة بالعمود القيمة Automatic. |

لاحظ أنك ستواجه مشكلة لو غيرت قيمة هذه الخاصية إلى ColumnHeaderSelect أو FullColumnSelect في وقت التصميم.. تعال نجرب هذا بأنفسنا:

- ابدأ مشروعا جديدا اسمه SelectionMode.
- ضع على النموذج جدول عرض، واضغط F4 لعرض نافذة الخصائص.
- اضغط الرابط Add Column الموجود في الهامش السفلي لنافذة الخصائص.. سيفتح هذا نافذة إنشاء عمود جديد مباشرة.. غير خصائص العمود، واضغط Ok لإغلاق النافذة.
- اضغط الرابط Edit Columns الموجود في الهامش السفلي لنافذة الخصائص، لفتح نافذة محرر مجموعة الأعمدة.. غير قيمة الخاصية SortMode الخاصة بالعمود الذي أنشأته إلى NotSortable حتى لا يحدث خطأ عند السماح بتحديد العمود كله، واضغط Ok.

- حدد الخاصية SelectionMode في نافذة الخصائص، ومن القائمة المنسدلة اختر ColumnHeaderSelect.
- اضغط F5 لتشغيل البرنامج.. ستجد أن خطأ حدث في البرنامج منع عرض النموذج!

لعلك مندهش، وتهتف متعجبا: لقد فعلنا كل ما هو مطلوب، فلماذا حدث هذا الخطأ؟!

السبب في هذا، يكمن في الكود الذي يحفظ قيم الخصائص التي تغيرها في وقت التصميم.. هذا الكود مكتوب في الملف From1.Designer.vb، وهو مرتب بحيث يكتب خصائص جدول العرض أولا، تليها خصائص أعمدته.. هذا معناه أن الخاصية SelectionMode تتغير أولا إلى ColumnHeaderSelect، قبل أن تتغير الخاصية SortMode الخاصة بالعمود عن قيمتها الافتراضية، مما يسبب هذا الخطأ.. ولا أنصحك بتغيير الكود الموجود في هذا الملف يدويا، لأن ما ستفعله سيضيع هباء عند أول تعديل تجريه على النموذج في وقت التصميم، لأنه سيؤدي إلى إعادة إنتاج كود ملف التصميم!

لهذا ليس أمامك إلا حل واحد: أن تستخدم نافذة الخصائص لإعادة قيمة الخاصية SelectionMode إلى RowHeaderSelect، وتستخدم حدث تحميل النموذج لتغيير قيمة هذه الخاصية كالتالي:

```
DataGridView1.SelectionMode =
        DataGridViewSelectionMode.ColumnHeaderSelect
```

الآن لو شغلت البرنامج فسيعمل بشكل صحيح، وسيمكنك تحديد أي عمود بضغط رأسه بالفأرة.

### حرف الجدولة القياسي StandardTab:

إذا جعلت قيمة هذه الخاصية True، فسيؤدي ضغط زر الجدولة TAB من لوحة المفاتيح إلى الانتقال من جدول العرض إلى الأداة التالية له على النموذج.. والقيمة الافتراضية لهذه الخاصية False، لهذا يؤدي ضغط زر الجدولة إلى الانتقال بين خانات جدول العرض، وفي هذه الحالة يمكن أن يضغط المستخدم Ctrl+TAB للانتقال من جدول العرض إلى الأداة التالية له على النموذج.

### المؤشر المستخدم UserSetCursor:
تعيد كائن مؤشر الفارة Cursor.. وهي تختلف عن الخاصية Cursor الموروثة من الأداة الأم Control، في أنها تعيد قيمة مؤشر الفارة الأصلية، مهما تغير شكل المؤشر نتيجة مروره فوق بعض المناطق الخاصة من جدول العرض.

كما يمتلك جدول عرض البيانات الوسيلتين التاليتين:

### ترتيب Sort:
ترتب صفوف جدول العرض، ويمكنك استدعاءها في حدث ضغط رأس العمود ColumnHeaderMouseClick، للتحكم في كيفية ترتيب الصفوف.. ولهذه الوسيلة صيغتان:

١- الصيغة الأولى تستقبل كائن واجهة المقارنة IComparer الذي تريد استخدامه في عملية الترتيب (راجع كتاب برمجة إطار العمل).

٢- والصيغة الثانية تستقبل معاملين:
- كائن العمود DataGridViewColumn الذي سيتم ترتيب الصفوف تبعا لقيم خاناته.
- إحدى قيمتي المرقم ListSortDirection التي توضح إن كان الترتيب تصاعديا Ascending أم تنازليا Descending.

### اختبار الضغط HitTest:
تخبرك بمعلومات عن موضع معين في جدول العرض، ولها معاملان:
- الإحداثي الأفقي X للموضع.
- الإحداثي الرأسي Y للموضع.

وتعيد هذه الوسيلة كائنا من نوع فئة "معلومات اختبار الضغط" HitTestInfo، وهي فئة معرفة داخل الفئة DataGridView، تمتلك الخصائص التالية:

| | | |
|---|---|---|
| S | Nowhere | تعيد كائن معلومات اختبار الضغط HitTestInfo، يشير إلى نقطة موجودة في منطقة فارغة من جدول العرض (ليست بها خانات عادية أو خانات عناوين). |
| | Column Index | تعيد رقم العمود الذي توجد فيه نقطة الاختبار. |

| | | |
|---|---|---|
| تعيد الموضع الأفقي لحافة العمود الذي توجد فيه نقطة الاختبار. | ColumnX | |
| تعيد رقم الصف الذي توجد فيه نقطة الاختبار. | RowIndex | |
| تعيد الموضع الرأسي لحافة الصف الذي توجد فيه نقطة الاختبار. | RowY | |
| تعيد إحدى قيم المرقم DataGridView.HitTestType، التي تخبرك بنوع المنطقة التي توجد بها نقطة الاختبار.. وهذه القيم هي:<br>- None: منطقة فارغة.<br>- Cell: خانة.<br>- ColumnHeader: رأس عمود.<br>- RowHeader: رأس صف.<br>- TopLeftHeader: الخانة الرئيسية العلوية اليسرى.<br>- HorizontalScrollBar: المنزلق الأفقي.<br>- VerticalScrollBar: المنزلق الرأسي. | Type | |

وقد استخدمنا هذه الوسيلة في حدث حركة الفأرة MouseMove الخاص بجدول العرض في المشروع SelectionMode، لنعرض في اللافتة LbInfo معلومات عن النقطة التي تتحرك فوقها الفأرة.

كما يمتلك جدول عرض البيانات الأحداث التالية:

**اكتمال ربط البيانات DataBindingComplete:**

ينطلق بعد حدوث تغير في مصدر البيانات، أو بعد تغيير قيمة أي من الخصائص التالية: DataSource, DataMember, BindingContext..

هذا مفيد عندما تريد أداء بعض المهام تناسب التغير الذي حدث في البيانات التي يعرضها جدول العرض، مثل تغيير عرض بعض الأعمدة وغير ذلك.

والمعامل الثاني e من النوع DataGridViewBindingCompleteEventArgs، وهو يملك الخاصية ListChangedType التي تخبرك بنوع التغير الذي حدث في مصدر البيانات، وهي تعيد إحدى قيم المرقم ListChangedType، التي تعرفنا عليها عند شرح الحدث IBindingList.ListChanged في فصل عروض البيانات Data Views.

### خطأ في البيانات DataError:

ينطلق عند حدوث خطأ في مصدر البيانات، أو عند حدوث خطأ في تنسيق أو تحويل قيمة إحدى خانات الجدول.. والمعامل الثاني e لهذا الحدث من نوع الفئة DataGridViewDataErrorEventArgs، وهي ترث الفئة DataGridViewCellCancelEventArgs التي تعرفنا عليها سابقا، والتي تخبرك برقم العمود ورقم الصف اللذين توجد بهما الخانة التي سببت الخطأ.. وفي حالة حدوث الخطأ في مصدر البيانات الخارجي، فإن الخاصيتين e.ColumnIndex و e.RowIndex تخبرانك بموضع الخانة الحالية في جدول العرض، حتى لو لم تكن مسئولة عن الخطأ أو مرتبطة به بشكل مباشر.. كما يمكنك أيضا استخدام الخاصية e.Cancel لإلغاء مغادلاة الخانة التي سببت الخطأ لو كان الخطأ حدث بسبب تحرير إحدى الخانات.

وإضافة إلى هذه الخصائص الموروثة، يمتلك المعامل e الخصائص التالية:

| | | |
|---|---|---|
| | Context | تعيد إحدى قيم المرقم DataGridViewDataErrorContexts التي توضح سبب الخطأ، وقد تعرفنا عليها سابقا. |
| | Exception | تعيد كائن الاستثناء Exception الذي يحتوي معلومات عن الخطأ الذي حدث. |
| | Throw Exception | إذا جعلت قيمة هذه الخاصية True، فسيتم إطلاق الخطأ في البرنامج بعد انتهاء تنفيذ كود الحدث الحالي.. والقيمة الافتراضية لهذه الخاصية هي False. |

ويتصرف جدول العرض كالتالي إذا لم تكتب إجراء يستجيب لهذا الحدث:

١- عند مغادر صف توجد أخطاء بإحدى خاناته، يعرض جدول العرض رسالة خطأ افتراضية للمستخدم فيها تفاصيل أكثر من اللازم عن الخطأ، وهي رسالة قبيحة حقا ومنفرة!

٢- يلغي القيم التي سببت الخطأ، وينقل المؤشر إلى الصف الذي أراده المستخدم.. هذا مستفز جدا لأنه يزيل القيم التي أدخلها المستخدم ولو كان الخطأ في صف جديد فإنه يحذفه بالكامل!

وللتخلص من هذا الأداء الشنيع، استخدم الكود التالي في هذا الحدث:

```
e.Cancel = True
Beep( )
```

هذا سيحقق لك فائتين:

١- منع عرض رسالة الخطأ التلقائية، وتشغيل نغمة تحذير بدلا منها.
٢- إجبار المستخدم على البقاء في نفس الصف دون إلغاء أي قيم أدخلها، مما يتيح له تعديل أخطائه.

### انزلاق Scroll:

ينطلق عندما يحرك المستخدم أحد المنزلقين.. والمعامل الثاني e لهذا الحدث من النوع ScrollEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | OldValue | تعيد عددا صحيحا يدل على موضع المنزلق قبل تحريكه. |
| | NewValue | تعيد عددا صحيحا يدل على موضع المنزلق بعد تحريكه. |
| | Scroll Orientation | تعيد إحدى قيمتي المرقم ScrollOrientation لتخبرك بوضعية المنزلق الذي سبب الحدث.. وهاتان القيمتان هما:<br>- HorizontalScroll: المنزلق الأفقي.<br>- VerticalScroll: المنزلق الرأسي. |
| | Type | تعيد إحدى قيم المرقم ScrollEventType التي تخبرك بسبب انطلاق الحدث، وهذه القيم هي:<br>- SmallDecrement: تم تحريك المنزلق خطوة صغيرة إلى الخلف.<br>- SmallIncrement: تم تحريك المنزلق خطوة صغيرة إلى الأمام.<br>- LargeDecrement: تم تحريك المنزلق قفزة كبيرة إلى الخلف.<br>- LargeIncrement: تم تحريك المنزلق قفزة كبيرة إلى الأمام.<br>- ThumbPosition: تم تحريك مؤشر الانزلاق إلى الأمام أو الخلف.<br>- ThumbTrack: يتم الآن تحريك مؤشر الانزلاق إلى الأمام أو الخلف.<br>- EndScroll: توقفت عملية الانزلاق.<br>- First: وصل المنزلق إلى أقل قيمة له.<br>- Last: وصل المنزلق إلى أقصى قيمة له. |

## تغير التحديد SelectionChanged:

ينطلق عندما تتغير الخانات أو الصفوف أو الأعمدة المحددة.

## مقارنة الترتيب SortCompare:

ينطلق عند مقارنة قيمتي خانتين في أحد الأعمدة أثناء عملية ترتيب الصفوف، وهو لا ينطلق إذا كان جدول العرض مرتبطا بمصدر بيانات، أو كان جدول العرض في الوضع الافتراضي VirtualMode.

والمعامل الثاني e لهذا الحدث من النوع ، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | Column | تعيد كائن العمود DataGridViewColumn الذي يتم ترتيبه. |
| | CellValue1 | تعيد قيمة الخانة الأولى. |
| | CellValue2 | تعيد قيمة الخانة الثانية. |
| | RowIndex1 | تعيد رقم الصف الذي توجد فيه الخانة الأولى. |
| | RowIndex2 | تعيد رقم الصف الذي توجد فيه الخانة الثانية. |
| | SortResult | ضع في هذه الخاصية نتيجة المقارنة، وهي تأخذ ثلاث قيمة:<br>- عدد أصغر من الصفر إذا كانت الخانة الأولى تسبق الثانية في الترتيب.<br>- عدد أكبر من الصفر إذا كانت الخانة الثانية تسبق الأولى في الترتيب.<br>- صفرا إذا كانت الخانتان متساويتين. |
| | Handled | اجعل قيمة هذه الخاصية True، إذا أردت ألا يقوم جدول العرض بأية عمليات مقارنة للخانتين بعد هذا. |

ويسمح لك هذا الحدث بالتحكم في كيفية مقارنة خانات الجدول إذا كانت تحتوي على قيم مركبة تصعب مقارنتها مباشرة.. لكن لأهمية الفعلية لهذا الحدث تنبع من أنه يتيح لك الفرصة لترتيب صفوف الجدول تبعا لأكثر من عمود.. لكي تفعل هذا، اتبع الخوارزمية التالية:

١- قارن القيمتين CellValue1 و CellValue2، فإن كانت إحداهما أكبر من الأخرى، فضع ناتج المقارنة في الخاصية SortResult.

٢- إذا كانت القيمتان CellValue1 و CellValue2 متساويتين، فيمكنك استخدام عمود آخر للترتيب على أساسه، وليكن اسمه Col2.

3- استخدم الخاصيتين RowIndex1 و RowIndex2 لقراءة قيمتي الخانتين الموجودتين في العمود Col2، وقارن بينهما، وضع ناتج المقارنة في الخاصية SortResult.

4- إذا تساوت القيمتان من العمود Col2 أيضا، فيمكنك استخدام عمود ثالث للترتيب بنفس الطريقة، إذا كنت ترى هذا ملائما.

والكود التالي يوضح لك هذه الخطوات:

```
If e.CellValue1.ToString < e.CellValue2.ToString Then
   e.SortResult = -1
ElseIf e.CellValue1.ToString > e.CellValue2.ToString Then
   e.SortResult = 1
Else ' سنقارن خانتين من عمود آخر
   Dim V1 As String = DataGridView1("Col2",
         e.RowIndex1).Value.ToString( )
   Dim V2 As String = DataGridView1("Col2",
         e.RowIndex2).Value.ToString( )
   If V1 < V2 Then
      e.SortResult = -1
   Else
      e.SortResult = 1
   End If
End If
e.Handled = True
```

**تم ترتيبه Sorted:**

ينطلق بعد انتهاء ترتيب صفوف جدول العرض.

**التعامل مع جدول العرض في الوضع الافتراضي VirtualMode:**

يتيح لك جدول العرض التحكم في عرض البيانات به بطريقتك الخاصة، وهو ما يعرف بالوضع الافتراضي VirtualMode، وهو مماثل للوضع الافتراضي الخاص بقائمة العرض ListView التي تعرفنا عليها في كتاب "برمجة نماذج الويندوز".

وفي الوضع الافتراضي، لا يحتفظ جدول العرض إلا بجزء محدود من البيانات جاهزة في الذاكرة، بينما يترك لك مسئولية إمداده بقيم الخانات التي يحتاج لعرضها على الشاشة كلما تحرك المستخدم بالمنزلق الأفقي أو الرأسي.. هذا مفيد في الحالات التالية:

١- عند التعامل مع مصادر بيانات لا يمكن ربطها مباشرة بجدول العرض، كالملفات الثنائية أو مجموعة مختلفة من المصفوفات أو غير ذلك.

٢- عند الحاجة إلى عرض كم هائل من البيانات، ففي الوضع الافتراضي لا يستهلك جدول العرض إلا جزءا محدودا من الذاكرة مهما زاد حجم البيانات التي تريد عرضها، لأنه فعليا لا يحتفظ إلا بالجزء الذي يعرضه على الشاشة.. لهذا يكون الوضع الافتراضي أكفأ وأسرع في عرض البيانات الضخمة.

٣- عندما لا يكون هناك مصدر بيانات، وإنما يتم توليد البيانات بناء على معادلة أو شرط أو ما شابه.. في هذه الحالة يوفر عليك الوضع الافتراضي عناء ملء مصفوفة أو قاعدة بيانات بالبيانات المولدة، ثم ربطها بجدول العرض كمصدر بيانات، ففي الوضع الافتراضي يمكن توليد قيمة الخانة مباشرة في الحدث CellValueNeeded.. والمشروع VirtualModeSample يريك مثالا على هذا، ففيه يعرض الجدول خمسة أعمدة ومليون صف (أي خمسة ملايين خانة)، حيث يحتوي العمود الأول على الأعداد من ١ إلى مليون، بينما يحتوي العمود الثاني على مربع قيم العمود الأول، ويحتوي العمود الثالث على قيم العمود الأول أس ٣ وهكذا، وهو ما يمكن توليده بالمعادلتين التاليتين:

قيمة أي خانة في العمود الأول = رقم الصف الذي توجد به + ١.

قيمة أي خانة في أي عمود آخر = (رقم الصف الذي توجد به + ١) أس (رقم العمود الذي توجد به + ١).

والآن لو جربت تشغيل هذا المشروع، فستجد أنه لن يستغرق وقتا قبل أن يظهر النموذج، وعليه جدول العرض وقد احتوى على الخانات مليئة بالأرقام.. في الحقيقة لم يستغرق ذلك وقتا لأن جدول العرض حسب فقط قيم الخانات التي تراها أمامك، وعندما تسحب المنزلق الرأسي إلى أسفل،

فسيقوم بحساب قيم الخانات الجديدة التي ستظهر لك.. هذا يجعل التعامل مع خمسة ملايين خانة عملية في غاية السرعة والكفاءة!

ويمنحك جدول العرض العناصر التالية للتعامل معه في الوضع الافتراضي:

### الوضع الافتراضي VirtualMode:

إذا جعلت قيمة هذه الخاصية True، فسيصير جدول العرض في الوضع الافتراضي، وعليك التحكم في كيفية عرض وتحديث قيم خاناته، وكيفية إضافة وحذف صفوفه.

### تحديث نص خطأ الصف UpdateRowErrorText:

تجبر جدول العرض على إطلاق الحدث RowErrorTextNeeded لتحديث نص الخطأ الخاص بصف معين، مما يتيح لك تغيير نص الخطأ بنفسك.. ولهذه الوسيلة صيغتان:

١. الصيغة الأولى تستقبل رقم الصف المراد تحديث نص خطئه.. ويمكنك استخدام -١ للإشارة إلى صف رؤوس الأعمدة.
٢. الصيغة الثانية تحدث نصوص الخطأ لنطاق من الصفوف، لهذا فهي تستقبل معاملين: رقم أول صف ورقم آخر صف في النطاق.

### تحديث معلومات ارتفاع الصف UpdateRowHeightInfo:

تجبر جدول العرض على إطلاق الحدث RowHeightInfoNeeded لتحديث ارتفاع صف معين، مما يتيح لك تغيير ارتفاع الصفوف بنفسك.. وتستقبل هذه الوسيلة معاملين:

- رقم الصف المراد تحديث ارتفاعه.. ويمكنك استخدام -١ للإشارة إلى صف رؤوس الأعمدة.
- معامل منطقي، إذا جعلته True فسيتم تحديث ارتفاع كل الصفوف التالية للصف الذي أرسلت رقمه إلى المعامل الأول.

### تحديث قيمة الخانة UpdateCellValue:

تجبر جدول العرض على إطلاق الحدث CellValueNeeded، لتحديث قيمة إحدى خانات جدول العرض.. وتستقبل هذه الوسيلة معاملين:

- رقم العمود الذي توجد به الخانة.. ويمكنك استخدام -١ للإشارة إلى عمود رؤوس الصفوف.
- رقم الصف الذي توجد به الخانة.. ويمكنك استخدام -١ للإشارة إلى صف رؤوس الأعمدة.

### تحديث نص خطأ الخانة UpdateCellErrorText:

تجبر جدول العرض على إطلاق الحدث CellErrorTextNeeded، لتحديث نص خطأ إحدى خانات جدول العرض.. ولها معاملان:

- رقم العمود الذي توجد به الخانة.. ويمكنك استخدام -١ للإشارة إلى عمود رؤوس الصفوف.
- رقم الصف الذي توجد به الخانة.. ويمكنك استخدام -١ للإشارة إلى صف رؤوس الأعمدة.

### قيمة الخانة مطلوبة CellValueNeeded:

ينطلق عندما يرسم جدول العرض إحدى خاناته، ويحتاج منك إلى إمداده بقيمتها.. في هذه الحالة عليك حساب قيمة الخانة كما فعلنا في التطبيق VirtualModeSample، أو الحصول على قيمة الخانة من الملف أو المصفوفة أو المصفوفة القائمة ArrayList التي تحتفظ فيها ببيانات الجدول.

والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellValueEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | ColumnIndex | تعيد رقم العمود الذي يحتوي على الخانة. |
| | RowIndex | تعيد رقم الصف الذي يحتوي على الخانة. |
| | Value | تعيد القيمة الموجودة حاليا في الخانة.. ويمكنك أن تضع في هذه الخاصية القيمة الجديدة التي تريد أن تعرضها الخانة. |

### دفع قيمة الخانة CellValuePushed:

ينطلق عندما يغير المستخدم قيمة إحدى خانات الجدول في وضعه الافتراضي، ليمكنك من حفظ القيمة الجديدة في وسيط التخزين الخاص بك (ملف أو مصفوفة أو غير ذلك).. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellValueEventArgs، كما في الحدث السابق.

### صف جديد مطلوب NewRowNeeded:

ينطلق بمجرد انتقال المستخدم إلى الصف الجديد (الصف الأخير) في جدول العرض في وضعه الافتراضي.. هذا يتيح لك وضع القيم الافتراضية في هذا الصف.. لاحظ أن تحرير المستخدم لأية خانة في هذا الصف سيستدعي الحدث CellValueNeeded لكل خانات الصف، ولو جربت هذا في المشروع VirtualModeSample، فستجد أن خانات الصف الجديد قد امتلأت بكل الأرقام المحسوبة.

## القيم الافتراضية مطلوبة DefaultValuesNeeded:

ينطلق عند إضافة صف جديد فارغ إلى نهاية جدول العرض، في وضعه الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.
والمعامل الثاني لهذا الحدث من النوع DataGridViewRowEventArgs الذي تعرفنا عليه سابقا، ويمكنك استخدام الخاصية e.Row لملء خانات الصف بالقيم الافتراضية، مع ملاحظة أن هذه القيم ستحفظ مباشرة في مصدر البيانات، أو ستؤدي إلى انطلاق الحدث CellValuePushed في الوضع الافتراضي لتتيح لك حفظها في وسيط التخزين الخاص بك.

## إلغاء تحرير الصف CancelRowEdit:

ينطلق عند إلغاء تحرير أحد صفوف جدول العرض في وضعه الافتراضي.
والمعامل الثاني e لهذا الحدث من النوع QuestionEventArgs، وهي يمتلك الخاصية المنطقية Response، التي إذا جعلتها True كان هذا معناه الموافقة على تنفيذ الحدث (إلغاء التحرير)، إما إذا جعلتها False كان هذا معناه رفض تنفيذ الحدث (وهذا معناه استمرار عملية التحرير وعدم إلغائها).

## الحالة القذرة للصف مطلوبة RowDirtyStateNeeded:

ينطلق عندما يريد جدول العرض معرفة إن كان الصف الحالي قد حفظ التغييرات التي حدثت في خاناته أم لا.. والمعامل الثاني e لهذا الحدث من النوع QuestionEventArgs، مع ملاحظة أنك إذا جعلت للخاصية e.Response القيمة True (وهي القيمة الافتراضية)، كان هذا معناه أن الصف لم يحفظ التغييرات بعد، لهذا سينطلق الحدث CancelRowEdit إذا حاول المستخدم إلغاء عملية التحرير بضغط الزر Esc.. إما إذا جعلت قيمة هذه الخاصية False، كان هذا معناه أن الصف قد حفظ التغييرات، ولن ينطلق الحدث CancelRowEdit.

## نص خطأ الصف مطلوب RowErrorTextNeeded:

ينطلق عندما يحتاج جدول العرض إلى النص الذي يحتوي على الأخطاء التي حدثت في الصف الحالي، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowErrorTextNeededEventArgs، وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| 🔒 | RowIndex | تعيد رقم الصف. |
| | ErrorText | ضع في هذه الخاصية نص الخطأ الخاص بالصف. |

**نص خطأ الخانة مطلوب CellErrorTextNeeded:**

ينطلق عندما يحتاج جدول العرض إلى النص الذي يحتوي على الأخطاء التي حدثت في الخانة الحالية، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellErrorTextNeededEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| 🔒 | ColumnIndex | تعيد رقم العمود الذي توجد به الخانة. |
| 🔒 | RowIndex | تعيد رقم الصف الذي توجد به الخانة. |
| | ErrorText | ضع في هذه الخاصية نص الخطأ الخاص بالخانة. |

**نص تلميح الخانة مطلوب CellToolTipTextNeeded:**

ينطلق عندما يحتاج جدول العرض إلى نص التلميح الخاص بإحدى خاناته، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellToolTipTextNeededEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| 🔒 | ColumnIndex | تعيد رقم العمود الذي توجد به الخانة. |
| 🔒 | RowIndex | تعيد رقم الصف الذي توجد به الخانة. |
| | ToolTipText | ضع في هذه الخاصية نص تلميح الشاشة الخاص بالخانة. |

**رف القائمة الموضعية للصف مطلوب RowContextMenuStripNeeded:**

ينطلق عندما يضغط المستخدم بزر الفأرة الأيمن على أحد الصفوف طالبا عرض القائمة الموضعية، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. هذا مفيد عندما تريد تغيير بعض عناصر القائمة الموضعية تبعا لمحتويات الصف أو حالته.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowContextMenuStripNeededEventArgs، وهو يمتلك الخاصيتين التاليتين:

| | | |
|---|---|---|
| | RowIndex | تعيد رقم الصف. |
| | Context MenuStrip | ضع في هذه الخاصية كائن رق القائمة الموضعية ContextMenuStrip التي تريد عرضها. |

### رف القائمة الموضعية للخانة مطلوب CellContextMenuStripNeeded:

ينطلق عندما يضغط المستخدم بزر الفأرة الأيمن على إحدى الخانات طالبا عرض القائمة الموضعية، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. هذا مفيد عندما تريد تغيير بعض عناصر القائمة الموضعية تبعا لمحتويات الخانة أو حالتها.
ولو لم تستجب لهذا الحدث، فإن الجدول سيعرض القائمة الموضعية التي استخدمتها في الحدث RowContextMenuStripNeeded.
والمعامل الثاني e لهذا الحدث من النوع DataGridViewCellErrorTextNeededEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | ColumnIndex | تعيد رقم العمود الذي توجد به الخانة. |
| | RowIndex | تعيد رقم الصف الذي توجد به الخانة. |
| | Context MenuStrip | ضع في هذه الخاصية كائن رق القائمة الموضعية ContextMenuStrip التي تريد عرضها. |

### معلومات ارتفاع الصف مطلوبة RowHeightInfoNeeded:

ينطلق عندما يحتاج جدول العرض إلى تغيير ارتفاع أحد الصفوف، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.. هذا مفيد عندما تريد تغيير ارتفاع بعض الصفوف بعد ترتيبها.. والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowHeightInfoNeededEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | RowIndex | تعيد رقم الصف. |
| | Height | ضع في هذه الخاصية ارتفاع الصف. |
| | MinimumHeight | ضع في هذه الخاصية أقل ارتفاع مسموح به للصف. |

### دفع معلومات ارتفاع الصف RowHeightInfoPushed:

ينطلق عندما يغير المستخدم ارتفاع أحد الصفوف، وذلك عندما يكون جدول العرض في الوضع الافتراضي، أو عندما يكون مرتبطا بمصدر بيانات.

والمعامل الثاني e لهذا الحدث من النوع DataGridViewRowHeightInfoPushedEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| | RowIndex | تعيد رقم الصف. |
| | Height | تعيد ارتفاع الصف. |
| | MinimumHeight | تعيد أقل ارتفاع مسموح به للصف. |
| | Handled | إذا جعلت قيمة هذه الخاصية True، فلن يتغير ارتفاع الصف، وسيظل بنفس الارتفاع السابق قبل أن يحاول المستخدم تغييره. |

### تحسين أداء جدول العرض:

رأينا في هذا الفصل، كيف أن جدول العرض أداة غنية تمتلك قدرات هائلة.. للأسف، هناك عيب في هذا الأمر، يظهر عند عرض عدد ضخم من السجلات في جدول العرض، فكل تلك الفئات التي تتعامل مع الأعمدة والصفوف والخانات بكل ما تحتويه من خصائص، تحتاج إلى مساحة كبيرة في الذاكرة، مما يقلل من كفاءة جدول العرض عند التعامل مع عدد ضخم من السجلات.. في هذه الحالة مثلا، قد يكون تحريك المنزلق لعرض صفوف معينة عملية ثقيلة وبطيئة، وكذلك عرض قائمة موضعية، وما إلى ذلك، إضافة إلى بطء البرنامج ككل بسبب العبء على الذاكرة!

ولحسن الحظ، لم يقف مصممو هذه الأداة العملاقة عاجزين أمام هذا العيب، فوضعوا بعض المعايير التي تساعد على تحسين أداء جدول العرض عند التعامل مع عدد ضخم من الصفوف والأعمدة.. ومنها:

١- لتغيير شكل الخانات، استخدم الخاصية DataGridView.DefaultCellStyle، فأي تغيير في هذه الخاصية ترثه جميع الخانات التي ليس لها طراز خاص بنفسها، مما يجعل استهلاك الذاكرة أقل ما يمكن مهما كان عدد الخانات هائلا.. ولا تلجأ إلى تغيير طراز كل خانة على حدة من خلال الخاصية DataGridViewCell.Style إلا في أضيق الحدود، لأن هذا لأن كائن طراز كل خانة يستهلك مساحة من الذاكرة.

٢- لا تغير طراز الخانة من خلال قالب الصف DataGridView.RowTemplate، لأن هذا سينشئ نسخة خاصة من كائن الطراز لكل صف في الجدول، وسيكون هذا عبئا كبيرا إذا كان في الجدول آلاف الصفوف.

٣- إذا كانت بعض الخانات تعرض البيانات بتنسيق خاص، فلا تغير التنسيق من كائن الطراز الخاص بالصفوف أو الأعمدة أو الخانات، بل استخدم الحدث CellFormatting لتغيير تنسيق الخانة عند عرضها.. هذا سيوفر مساحة الذاكرة، ولن يستهلك وقتا ملموسا في التنفيذ، لأن هذا الحدث ينطلق فقط للخانات التي تظهر للمستخدم على الشاشة، وليس كل الخانات.

٤- عند قراءة طراز الخانة، استخدم الخاصية InheritedStyle بدلا من الخاصية Style، لأن الخاصية Style تنشئ كائن طراز جديد إذا حاولت قراءتها وهي فارغة!

٥- استخدم الخاصية ContextMenuStrip لوضع قائمة موضعية لجدول العرض ككل، ولا تضع قائمة موضعية لكل صف أو عمود أو خانة على حدة.

٦- لا تضع قائمة موضعية لقالب الصف DataGridView.RowTemplate، لأن هذا سينشئ نسخة من القائمة الموضعية لكل صف في الجدول.

٧- إذا كنت تحتاج إلى قائمة موضعية مختلفة لكل صف، فاستخدم الحدث RowContextMenuStripNeeded لعرض القائمة الموضعية للصف عند الحاجة.

٨- إذا كنت تحتاج إلى قائمة موضعية مختلفة لبعض الخانات، فاستخدم الحدث CellContextMenuStripNeeded لعرض القائمة الموضعية للخانة عند الحاجة.

٩- لا تستخدم التحجيم التلقائي Auto-Sizing على مستوى جدول العرض إذا كان يحتوي على عدد هائل من الخانات، لأن حساب أنسب عرض وارتفاع للصفوف والأعمدة، يستلزم إجراء عمليات قياس لأبعاد محتويات جميع الخانات!.. وإذا كان التحجيم التلقائي مهما لك، فيمكنك أن تنفذه على الخانات المعروضة فقط، وذلك بوضع القيمة DisplayedCells أو القيمة DisplayedCellsExceptHeaders في الخاصية AutoSizeMode، وإذا أردت تحجيم رءوس الصفوف تلقائيا فاستخدم القيمة AutoSizeToDisplayedHeaders، أو AutoSizeToFirstHeader.. لكن الأفضل على الإطلاق، منع التحجيم

التلقائي نهائيا، وقيامك بتنفيذه برمجيا، بكتابة الكود الذي يقيس عرض وارتفاع محتويات الخانات في الأحداث التالية:

- UpdateRowHeightInfo.
- Scroll.
- RowDividerDoubleClick.
- ColumnDividerDoubleClick.

١٠- عند التعامل مع الأعمدة والصفوف والخانات المحددة، استخدم المجموعات SelectedColumns و SelectedRows و SelectedCells بحذر، لأن كفاءتها تكون سيئة إذا كان الجدول يحتوي على عدد ضخم من الخانات.

١١- استخدم الوسيلة DataGridView.GetCellCount لمعرفة عدد الخانات المحددة، بدلا من استخدام الخاصية SelectedCells.Count.

١٢- استخدم الوسيلة Rows.GetRowCount لمعرفة عدد الصفوف المحددة، بدلا من استخدام الخاصية SelectedRows.Count.

١٣- استخدم الوسيلة Columns.GetColumnCount لمعرفة عدد الأعمدة المحددة، بدلا من استخدام الخاصية SelectedColumns.Count.

١٤- تجنب السماح للمستخدم بتحديد الخانات بصورة منفردة، وبدلا من هذا ضع في الخاصية SelectionMode القيمة FullRowSelect أو FullColumnSelect.

١٥- حافظ على الصفوف المشتركة Shared Rows بقدر الإمكان.. ولكن.. ما هي الصفوف المشتركة؟.. هذا هو موضوع الفقرة التالية.

**الصفوف المشتركة Shared Rows:**

فكرة هذه التقنية بسيطة، فكل صف جديد يتم إنشاؤه في جدول العرض يأخذ خصائصه الشكلية من قالب الصفوف RowTemplate، لهذا لا داعي لأن نحجز له مساحة كاملة في الذاكرة لنكرر فيها نفس البيانات المشتركة مع القالب.. هذا يوفر مساحة كبيرة في الذاكرة، خاصة إذا كان عدد صفوف جدول العرض ضخما.. وتلغى مشاركة الصف إذا استخدمت أية خاصية لتغيير طريقة عرضه.. بل إن مجرد تعامل المستخدم مع أي خانة في الصف يلغي مشاركته!

لهذا لا تفيدك تقنية مشاركة الصفوف، إلا إذا كان جدول البيانات يحتوي على عدد هائل من الخانات، ولا يتوقع أن يتعامل المستخدم مباشرة إلا مع عدد قليل منها.

كما أن هذه التقنية غير مفيدة إذا كان جدول العرض لا يرتبط بمصدر بيانات، لأن وضع أي قيمة في الخانة يلغي مشاركة الصف، وهذا منطقي، لأن حفظ هذه

القيمة في الذاكرة يحتاج إلى إنشاء كائن الخانة، وبالتالي كائن الصف الذي توجد به!. بينما في وجود مصدر بيانات خارجي ــ سواء من خلال تقنية الربط Binding أو من باستخدام الوضع الافتراضي Virtual Mode ــ لا يحتاج جدول العرض إلى حفظ القيم في الذاكرة، فهو يجلبها من مصدر البيانات عند الحاجة، ويرسمها في الخانات مباشرة.

| **وتنص قاعدة المشاركة على أنه:** |
| --- |
| يمكن مشاركة الصف فقط إذا كان من الممكن معرفة خصائص كل خانة من خاناته من خصائص الصف والعمود اللذين توجد فيهما. |

ويتم إلغاء مشاركة الصف إذا تغيرت حالة خانته، بحيث لا يعود من الممكن استنتاج خصائصها من خصائص الصف والعمود.. وهذه بعض الأمثلة على الحالات التي تلغى فيها مشاركة الصف:

- تحديد خانة منفردة في الصف، دون أن تكون في عمود محدد.. لو أردت منع هذه الحالة، فلا تسمح للمستخدم بتحديد خانة منفردة.
- وضع قيمة في الخاصية ToolTipText أو ContextMenuStrip لإحدى خانات الصف.. ويمكنك تجاوز هذه الحالة، باستخدام الحدثين CellToolTipTextNeeded و CellContextMenuStripNeeded لعرض تلميح الشاشة والقائمة الموضعية عند الحاجة إليهما.
- وجود قائمة منسدلة في إحدى خانات الصف، ووضع عناصر في الخاصية Items الخاصة بها بدون استخدام تقنية الربط.

ويمكنك استخدام الإرشادات التالية لإنشاء الصفوف المشتركة والمحافظة عليها في حالة المشاركة:

١- تجنب استخدام الصيغة التي تستقبل مصفوفة من القيم عند إضافة صف جديد إلى مجموعة الصفوف Rows باستخدام الوسيلة Add أو Insert، لأن هذه الصيغة تضع القيم في خانات الصف مما يلغي مشاركته.
لاحظ أن الصف الجديد الموجود في نهاية جدول العرض هو صف غير مشترك.

٢- تجنب التعامل مع الصف من خلال مجموعة الصفوف Rows، وتجنب المرور عبر الصفوف باستخدام حلقة التكرار For Each.. وبدلا من هذا استخدم الوسائل البديلة التي تستقبل رقم الصف للتعامل معه، مثل الوسائل GetContextMenuStrip، GetErrorText، GetPreferredHeight، GetState.

٣- تجنب التعامل مع الخانة من خلال المجموعة DataGridViewRow.Cells لأن هذا يلغي مشاركة الصف الذي توجد فيه.

٤- تعامل مع خصائص الصفوف والخانات من خلال الخصائص التي يمنحها لك المعامل e في الأحداث التي تنطلق عند حدوث تغييرات في الصف أو الخانة، فهذه الخصائص لا تلغي مشاركة الصف.

٥- استخدم الخاصية CurrentCellAddress للحصول على رقم العمود ورقم الصف اللذين توجد فيهما الخانة، فهذا لا يلغي مشاركة الصف.

٦- استخدم الخاصية Rows.SharedRow للحصول على كائن الصف دون إلغاء مشاركته.. ويمكنك إجراء التعديلات على هذا الكائن، لكن مع ملاحظة أن هذه التغييرات ستؤثر على كل الصفوف المشتركة معه!

٧- استخدم الوسيلة GetContextMenuStrip للحصول على القائمة الموضعية للصف المشترك، لأن الخاصية ContextMenuStrip الخاصة بالصف المشترك ستجد أن رقمه -١ ولن تعيد القائمة الموضعية بصورة صحيحة.

٨- لا تستخدم الحدثين CollectionChanged و RowStateChanged الخاصين بمجموعة الصفوف، لأنهما يلغيان مشاركة الصفوف.

٩- إذا كانت للخاصية SelectionMode القيم FullColumnSelect أو ColumnHeaderSelect أو FullRowSelect أو RowHeaderSelect فلا تتعامل مع الصفوف المحددة من خلال الخاصية DataGridView.SelectedCells لأنها ستلغي مشاركة الصفوف المحددة في هذه الحالة!

١٠- إذا كانت للخاصية SelectionMode القيمة CellSelect فلا تستخدم الوسيلة DataGridView.SelectAll، لأنها ستلغي مشاركة الصفوف.

١١- لا تستخدم الوسيلة DataGridView.AreAllCellsSelected لأنها تلغي مشاركة الصفوف!

١٢- لا تضع القيمة False في الخاصية ReadOnly أو Selected الخاصة بالخانة، إذا كان لأي من هاتين الخاصيتين القيمة True في العمود الذي توجد فيه الخانة.

١٣- لا تستخدم الخاصية DataGridView.Rows.List لأنها تلغي مشاركة جميع الصفوف!

١٤- لا تستخدم الصيغة DataGridView.Sort(IComparer) لوسيلة الترتيب، لأن استخدام فئة مقارنة خاصة يلغي مشاركة جميع الصفوف، بسبب حاجة فئة المقارنة إلى التعامل مع نسخ من الصفوف التي تقارنها.

١٥- استخدم الحدث RowUnshared أثناء تصميم البرنامج لرصد الحالات التي تلغي مشاركة الصفوف، وحاول تجنبها.

وللتأكد من أن الصف مشترك، استخدم الوسيلة SharedRow الخاصة بمجموعة الصفوف للحصول على كائن الصف، ومن ثم افحص رقم هذا الصف، فإن كان -١ فهذا معناه أنه صف مشترك، فكما ذكرنا سابقا، الصف المشترك رقمه دائما -١!..
والمثال التالي يخبرك إن كان الصف الأول مشتركا أم لا:

```
If DataGridView1.Rows.SharedRow(0).Index = -1 Then
   MsgBox("هذا الصف مشترك")
End If
```

أخيرا: أحتاج إلى تذكيرك إلى أن كل هذه المحاذير، تتعلق فقط بالحالة التي تتعامل فيها مع جدول بيانات يحتوي على عدد هائل من الصفوف يقدر بالآلاف.. وإن شئت نصيحتي، عليك الهروب من هذه الحالة المعقدة لأنها أساسا غير عملية، فلا يمكن للمستخدم أن يستعرض آلاف السجلات دفعة واحدة.. لهذا أنصح باستخدام تقنية أخرى، هي تقنية التقسيم إلى صفحات Paging.

**تقسيم جدول العرض إلى صفحات Paging:**

مَن تعامل مع جدول العرض في تطبيقات مواقع الإنترنت ASP.NET Web Applications، يعرف أن جدول العرض الخاص بها يسمح بعرض البيانات في صورة صفحات، كل منها تحتوي على عدد من السجلات (٢٠ أو ٣٠ مثلا)، ويعرض الجزء السفلي من جدول العرض أرقام الصفحات المتاحة في صورة روابط، وعند ضغط أي منها، يتم عرض السجلات المناظرة لهذه الصفحة في جدول العرض.

ولا أدري لماذا لم تقدم ميكروسوفت هذه التقنية البسيطة والجميلة في جدول العرض الخاص بتطبيقات الويندوز، فهي أسهل وأكفأ وأكثر ملاءمة للمستخدم من تقنية الصفوف المشتركة!

لهذا، دعنا ننشئ نحن بأنفسنا هذه التقنية.. الأمر بسيط، فكل المطلوب هو أن ننشئ موصل جدول له معاملان: الأول يستقبل رقم السجل والثاني يستقبل عدد السجلات المطلوبة، وذلك للحصول على عدد معين من سجلات الجدول بدءا من موضع معين.. وسنضع تحت جدول العرض عددا من لافتات الوصلات LinkLable لنعرض فيها أرقام الصفحات، وعند ضغطها نحمّل السجلات المطلوبة من قاعدة البيانات إلى مجموعة البيانات، ومن ثم نعرض هذه السجلات في جدول العرض من خلال تقنية الربط.. بهذه الطريقة سنحصل على وفر هائل في الذاكرة، ليس فقط بسبب قلة عدد سجلات جدول العرض، ولكن أيضا بسبب قلة سجلات مجموعة البيانات، فحتى لو كانت تقنية مشاركة الصفوف تقلل من

مساحة الذاكرة التي يستهلكها جدول العرض، إلا أنها لا تفعل شيئا حيال حجم الذاكرة التي تستهلكها مجموعة البيانات!.. هذا إضافة إلى سرعة تحميل البيانات من قاعدة البيانات، بسبب تقسيمها إلى أجزاء صغيرة.

وعليك اختيار عدد مناسب من السجلات لعرضه في كل صفحة.. ربما يكون العدد ٢٥ مناسبا لتطبيقات الويندوز، فهو عدد معقول بالنسبة للمستخدم، ولا يمثل عبئا ضخما على الذاكرة.. وعموما، لقد عرفنا الثابت RowsNo على مستوى النموذج، ويمكنك تعديله بسهولة للتعامل مع العدد الذي يناسبك من السجلات.

والآن، دعنا نرى كيف ننفذ هذه الفكرة:

- ابدأ مشروعا جديدا اسمه DataGridViewPaging.. وستجده مرفقا بأمثلة الكتاب.
- من القائمة العلوية Data اضغط الأمر Add New Data Source، واتبع خطوات المعالج السحري لإضافة جدولي المؤلفين والكتب إلى مصدر البيانات.
- افتح مخطط قاعدة البيانات واضغط موصل جدول المؤلفين AuthorsTableAdapter بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Configure، وعدل الاستعلام ليصير كالتالي:

```
SELECT * FROM dbo.Authors
WHERE ID BETWEEN @StartID AND @EndID
```

واضغط زر الموافقة.. سيعدل هذا الوسيلة Fill بإضافة معاملين لها، أحدهما اسمه StartID والآخر اسمه EndID، وبهذا يتم تحميل السجلات المحددة فقط من جدول المؤلفين.

- اضغط موصل جدول المؤلفين AuthorsTableAdapter بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Add Query، وتابع خطوات المعالج السحري لإضافة استعلام يعيد قيمة منفردة، باستخدام جملة SQL التالية:

```
SELECT MAX(ID) FROM Authors
```

وسمِّ الدالة التي تنفذ هذا الاستعلام GetMaxID.. هذه الدالة ستخبرنا برقم آخر مؤلف في جدول المؤلفين، لنستخدمه في معرفة عدد الصفحات اللازمة لعرض كل المؤلفين.

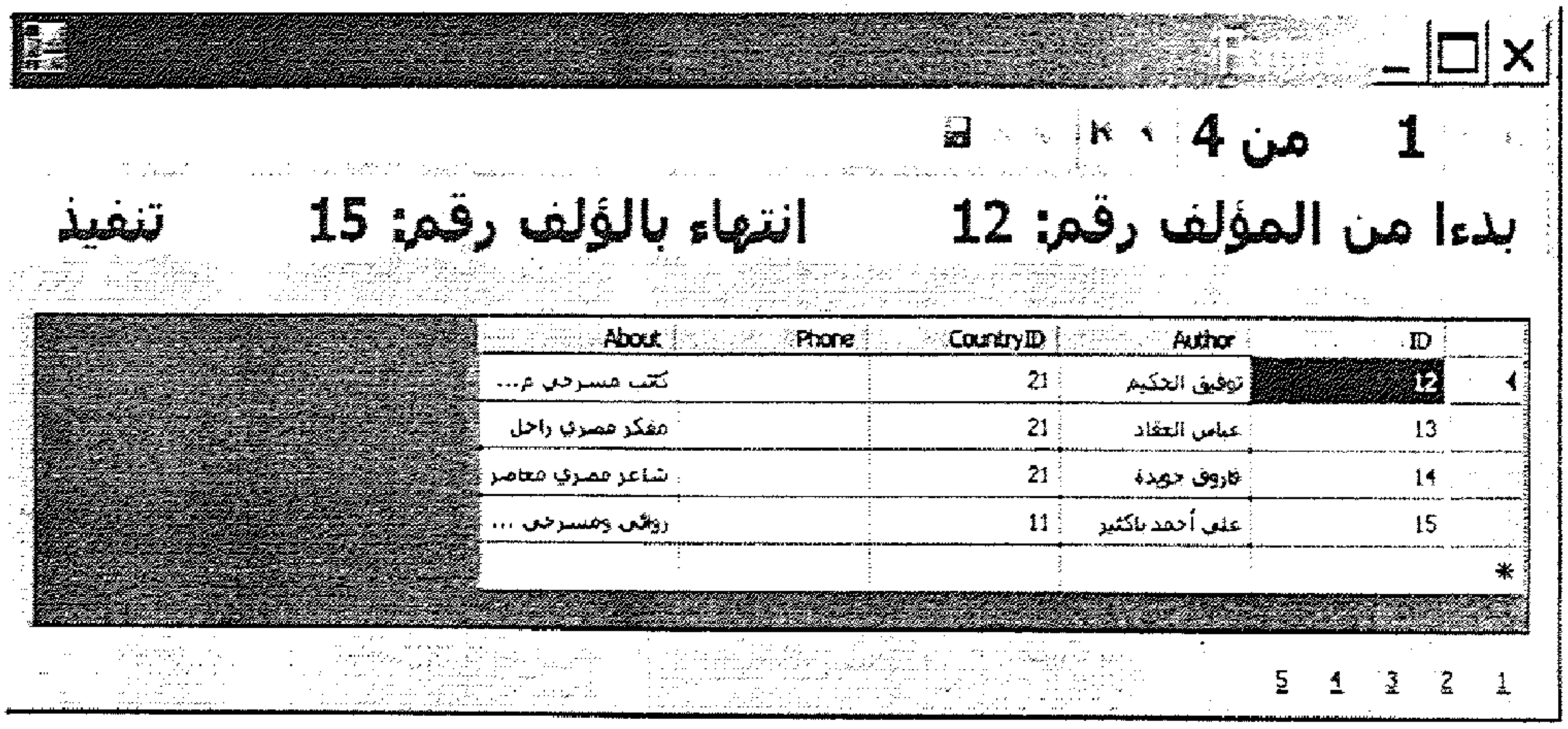
- اضغط موصل جدول الكتب BooksTableAdapter بزر الفأرة الأيمن، ومن القائمة الموضعية اضغط الأمر Configure، وعدل الاستعلام ليصير كالتالي:

```
SELECT Books.* FROM dbo.Books, Authors
WHERE AuthorId = Authors.ID
```

**AND Authors.ID BETWEEN @StartID AND @EndID**

واضغط زر الموافقة.. سيعدل هذا الوسيلة Fill بإضافة معاملين لها، أحدهما اسمه StartID والآخر اسمه EndID، وبهذا يتم تحميل كتب المؤلفين الذين نتعامل معهم حاليا فقط.

- لاحظ أن موصل البيانات لن ينشئ أوامر التحديث والإدراج والحذف الخاصة بجدول الكتب تلقائيا بسبب وجود عملية ربط في استعلام التحديد.. لهذا يتعين عليك إنشاء هذه الأوامر بنفسك إن كنت تريدها.
- انتقل إلى النموذج، ومن القائمة العلوية Data اضغط الأمر Show Data Sources لعرض نافذة مصادر البيانات.. اسحب جدول المؤلفين وأسقطه على النموذج.. سيضيف هذا إلى النموذج جدول عرض والأدوات اللازمة لربطه بجدول المؤلفين، مع وضع رف أدوات علوي، يسمح للمستخدم بإدخال قيمتي المعاملين StartID و EndID مع زر تنفيذ عملية الملء عنوانه Fill.. هذا جميل.. يمكنك أن تترك هذه الإمكانية أيضا للمستخدم، ليحدد بنفسه السجلات التي يريد عرضها، لكن مع تغيير عناوين اللافتات لتصير عربية كما في الصورة:




- لا تنس استخدام محرر مجموعة الأعمدة من نافذة الخصائص لإخفاء العمود RowVersion أو تغيير نوعه من عمود صورة إلى عمودي نصي، حتى لا يسبب أخطاء.. استخدم نافذة الخصائص أيضا، لإضافة عمود إلى الجدول يعرض أزرارا وامنحه الاسم ColBooks.
- أضف نموذجا آخر إلى المشروع اسمه FrmBooks، وضع عليه جدول عرض لنعرض فيه كتب المؤلف.
- انقر جدول العرض مرتين بالفأرة لعرض كود الحدث CellContentClick، واكتب فيه الكود الذي يعرض النموذج

FrmBooks ويعرض كتب المؤلف الحالي فيه.. لقد فعلنا هذا من قبل في المشروع DataGridViewAuthorBooks ولن نكرر شرحه هنا.

- ضع نسخة من موصل بيانات الكتب على النموذج، وسمّه BooksTableAdapter، لاستخدامه في ملء جدول الكتب بكتب المؤلفين المعروضين في الصفحة الحالية.. لاحظ أنك تستطيع إحضار كتب كل المؤلف من قاعدة البيانات مباشرة عند الحاجة إلى عرضها، لكن هذا سيزيد من عدد مرات الاتصال بقاعدة البيانات أثناء تعامل المستخدم مع المؤلفين المعروضين في الصفحة الحالية.. لهذا من الأفضل أن نحضر كل كتب هؤلاء المؤلفين مباشرة ونحفظها في مجموعة البيانات.. ونظرا لأن كل صفحة ستحتوي على ٢٥ مؤلفا، ومع افتراض أن لكل مؤلف ١٠ كتب في المتوسط، فإننا سنضع في الذاكرة حوالي ٢٥٠ كتابا. هذا رقم معقول ولا يقلقنا.. وعلى كل حال، أنت المسئول من خلال التجربة والخطأ، عن تحديد أي من الطريقتين تجعل برنامجك يعمل أسرع دون أن يخنق الخادم أو جهاز المستخدم.
- ضع لافتة رابط LinkLabel على النموذج وامنحها الاسم LnkPage1، وضع فيها النص "١"، واعرضها أسفل الجدول كما هو واضح في الصورة السابقة.. اضبط خط هذه اللافتة وثبت حافتيها اليمنى والسفلي باستخدام الخاصية Anchor.. سنستخدم هذه اللافتة كقالب نستمد منه خصائص باقي لافتات الرابط التي سنضيفها في وقت التشغيل.
- انقر لافتة الرابط مرتين بالفأرة لكتابة كود الحدث LinkClicked.. كود هذا الحدث بسيط للغاية، فنحن نستطيع حساب رقم أول مؤلف نريد عرضه، من خلال الرقم الذي تعرضه اللافتة باستخدام المعادلة التالية:

رقم البداية = ١ + عدد الصفوف في الصفحة × (رقم الصفحة – ١ )

ونستطيع حساب رقم النهاية من المعادلة التالية:

رقم النهاية = رقم البداية + عدد الصفوف في الصفحة.

علما بأن رقم الصفحة، هو النص الذي تعرضه اللافتة الحالية.. لاحظ أن هذا الحدث سيستجيب لأكثر من لافتة رابط، لهذا سنستخدم المعامل Sender لمعرفة اللافتة التي ضغطها المستخدم.

الآن، صار من السهل أن نملأ جدولي المؤلفين والكتب بالسجلات.. هذا هو الكود:

```
Dim Lnk = CType(sender, LinkLabel)
Dim StartID = 1 + RowsNo * (CInt(Lnk.Text) - 1)
AuthorsTableAdapter.Fill(BooksDataSet.Authors,
        StartID, StartID + RowsNo)
```

```
BooksTableAdapter.Fill(BooksDataSet.Books, StartID,
            StartID + RowsNo)
```

- يتبقى لنا الآن كتابة كود حدث تحميل النموذج، لإنشاء لافتات الصفحات.. يجب أن نحسب أولا عدد الصفحات المطلوبة، وذلك بقسمة رقم آخر مؤلف على عدد الصفوف التي سنعرضها في كل صفحة، مع تقريب الكسر إلى أكبر عدد صحيح:

```
Dim MaxID = AuthorsTableAdapter.GetMaxID
Dim PagesNo = Math.Ceiling(MaxID / RowsNo)
```

بعد هذا سننشئ لافتات الرابط التي ستعرض هذه الصفحات، ووضع كل منها بجوار اللافتة التي تسبقها بمسافة كافية.. في البداية نحن نعرف أن اللافتة السابقة هي اللافتة LnkPage1، وبعد هذا يجب أن نحتفظ بكل لافتة نضيفها في متغير اسمه PrvLnk لنستخدمها في ضبط موضع اللافتة التالية لها.. علينا أيضا أن نضبط خط اللافتة ونثبت حوافها، ونربطها بالحدث المستجيب لضغط الرابط.. هذا هو الكود:

```
Dim PrvLnk = LnkPage1
For I = 2 To PagesNo
    Dim Lnk As New LinkLabel
    Me.Controls.Add(Lnk)
    Lnk.Visible = True
    Lnk.Text = I
    Lnk.Font = PrvLnk.Font
    Lnk.AutoSize = True
    Lnk.Location = PrvLnk.Location -
                    New Point(Lnk.Width + 10, 0)
    Lnk.Anchor = PrvLnk.Anchor
    AddHandler Lnk.LinkClicked,
                    AddressOf LnkPage1_LinkClicked
    PrvLnk = Lnk
Next
```

لاحظ أن الخاصية AutoSize ستكون بلا تأثير إذا وضعت فيها True قبل أن تضيف لافتة الربط إلى أدوات النموذج.. لهذا أضفنا اللافتة إلى أدوات النموذج أولا، وضبطنا خطها، ووضعنا في الخاصية Text النص الذي ستعرضه، قبل أن نجعلها تغير حجمها تلقائيا لتناسب محتوياتها.

والآن، سيكون من الأفضل لو ضغطنا نحن رابط أول لافتة لنعرض أول صفحة من صفحات المؤلفين في جدول العرض بمجرد تشغيل البرنامج..

لكن نظرا لأن لافتة الربط لا تملك الوسيلة PerformClick الخاصة بالأزرار، فسنضطر إلى استدعاء الحدث LnkPage1_LinkClicked بأنفسنا كالتالي:

```
LnkPage1_LinkClicked(LnkPage1,
    New LinkLabelLinkClickedEventArgs(Nothing))
```

لو شغلت البرنامج الآن، فسيعرض جدول العرض أول صفحة من صفحات المؤلفين، ويمكنك أن تضغط رابط أي صفحة أخرى لعرضها.. ولو ضغطت زر عرض كتب أي مؤلف، فسيظهر النموذج الثاني وعليه كتب هذا المؤلف.

يبدو كل شيء جيدا.. لكن للأسف هناك مشكلة صغيرة، تحدث بسبب طريقة تقسيم الصفحات التي نستخدمها!.. فنحن لا نضمن انتظام أرقام المؤلفين لأن الحقل ID مولد تلقائيا.. لهذا قد تجد أن أول مؤلف يبدأ بالرقم ١٢ وليس ١، ويليه مؤلف رقمه ١٥ مثلا، وذلك بسبب حذف سجلات أخرى ضيعت الترقيم الوسيط!.. لهذا قد نجد صفحات تعرض عددا من السجلات أقل من العدد المطلوب، وأحيانا قد تظهر صفحات ليس فيها أية سجلات على الإطلاق!

ولحل هذه المشكلة، علينا تغيير طريقة تقسيم الصفحات، وهو ما فعلناه في المشروع DataGridViewPaging2.. في هذا المشروع تركزت معظم التغييرات على استعلامات موصلات الجداول، مع قليل من التعديلات في الكود.

دعنا نفهم فكرة التقسيم الجديدة:

في البداية لو أردنا ملء أول صفحة بعدد من المؤلفين @Count، فسنستخدم الاستعلام التالي، وهو الذي ستجده في الوسيلة FillFirstPage:

```
SELECT TOP (@Count) * FROM Authors
```

وسيتم ملء كتب هؤلاء المؤلفين بالوسيلة FillFirstPage في موصل جدول الكتب بالاستعلام التالي:

```
SELECT *
FROM  Books INNER JOIN
      Authors ON Books.AuthorID = Authors.ID
WHERE Authors.ID IN
(SELECT TOP (@Count) ID FROM Authors)
```

لاحظ أن هذا الاستعلام مركب، فهو يستخدم جملة SELECT ثانية للحصول على أرقام المؤلفين المعروضين في أول صفحة، والتأكد أن رقم المؤلف الذي نحصل على كتبه يقع ضمن هذه الأرقام.

هذا جميل.. لكن كيف نحصل على المؤلفين في الصفحات الأخرى غير الصفحة الأولى؟

المشكلة هنا أننا لا نعرف ترقيم أول مؤلف في هذه الصفحات، فكما ذكرنا من قبل، يتسم حقل الترقيم التلقائي بعدم الانتظام!

لحل هذه المشكلة، علينا معرفة ترقيم آخر مؤلف تم عرضه في الصفحة السابقة للصفحة الحالية.. افترض أن الصفحة التي سنعرضها ستبدأ بالمؤلف الحادي عشر.. هذا معناه أن الصفحات السابقة عرضت ١٠ مؤلفين.. يمكننا إذن أن نحصل على أرقام أول ١٠ مؤلفين من الجدول كالتالي:

```
SELECT TOP (11 - 1)
ID FROM   Authors AS PrvPages
ORDER BY ID
```

ويمكننا أن نحصل على ترقيم آخر مؤلف منهم باستخدام الدالة MAX كالتالي:

```
SELECT MAX(ID)
FROM  (
          SELECT TOP (11 -1)
          ID FROM   Authors AS PrvPages
          ORDER BY ID
         ) AS MaxID
```

طبعا الرقم ١١ يفيدنا في شرح هذا المثال، لكن في البرنامج، سنضع بدلا منه معاملا اسمه @AuthorNo وهو الرقم الفعلي للمؤلف الذي يظهر في بداية الصفحة، وليس ترقيمه التلقائي الموجود في الحقل ID.. هذا الرقم يساوي:

١ + عدد الصفوف في الصفحة × (رقم الصفحة -١ ) كما شرحنا من قبل.

والآن، بعد أن حصلنا على ترقيم آخر مؤلف عرضناه في الصفحة السابقة، يمكننا أن نقرأ السجلات التي يزيد ترقيمها عن ترقيمه، ونأخذ منها فقط العدد @Count.. هذا هو الاستعلام الكامل:

```
SELECT TOP (@Count)
* FROM  Authors
WHERE  ID > (
                SELECT MAX(ID)
                FROM  (
                        SELECT TOP (@AuthorNo -1)
                        ID FROM   Authors AS PrvPages
                        ORDER BY ID
                       ) AS MaxID
                )
```

| ملحوظة: |
| --- |
| ستضاف جملة التحديد بعد أمر التحديث وأمر الإدراج، وهذا سيسبب أخطاء في البرنامج عند حفظ التغييرات، بسبب وجود معاملين لن يتم إرسال قيمتيهما.. لهذا عليك حذف جملة التحديد من أمر التحديث وأمر الإدراج.. يمكنك فعل هذا من الخيارات المتقدمة Advanced Options أثناء تنفيذ المعالج السحري، أو يمكنك فعله من نافذة الخصائص بعد انتهاء المعالج.. حدد موصل جدول المؤلفين، وافتح نافذة الخصائص وأسدل خصائص أمر التحديث UpdateCommand وغير قيمة الخاصية CommandText.. لاحظ أن أمر التحديد يوجد في سطر جديد، لهذا لن تستطيع التعامل معه في خانة القيمة لأنها تعرض سطرا واحدا فقط.. للتحايل على هذا، حدد السطر الظاهر (هذا هو أمر التحديث) وقصه Cut، ثم الصقه مرة ثانية Paste.. هكذا تكون قد تخلصت من أمر التحديد.. ويمكنك فعل نفس الشيء مع أمر الإدراج الموجود في الخاصية InsertCommand. |

ولكي نحصل على كتب هؤلاء المؤلفين، سنكون استعلاما يضمن أن ترقيم المؤلف الذي نقرأ كتبه يقع ضمن أرقامهم كالتالي:

```sql
SELECT * FROM  Books
INNER JOIN Authors
ON Books.AuthorID = Authors.ID
WHERE Authors.ID IN
      (
       SELECT TOP (@Count)
       ID FROM  Authors
       WHERE  ID >
                 (
                  SELECT MAX(ID)
                  FROM  (
                         SELECT TOP (@AuthorNo -1)
                         ID FROM   Authors AS PrvPages
                         ORDER BY ID
                        ) AS MaxID
                 )

      )
```

واضح أن هذا أعقد استعلام كتبناه حتى الآن.. لكن لا تدعه يربكك، فكل ما هو بعد الكلمة IN في هذا الاستعلام هو نفس الاستعلام الذي استخدمناه في موصل جدول المؤلفين، مع فارق واحد: أننا هنا نقرأ الحقل ID فقط وليس كل حقول جدول المؤلفين، لأننا نريد استخدام الحقل ID في جملة الشرط.

يمكنك الآن تجربة المشروع DataGridViewPaging2.. ستجده يعمل بشكل رائع، فكل صفحة فعلا تعرض عدد المؤلفين المطلوب بطريقة دقيقة، ولا يستثنى من هذا إلا آخر صفحة، التي قد تعرض عددا أقل من المؤلفين، بسبب عدم وجود مزيد من المؤلفين في الجدول.. وهذا منطقي وصحيح.

مبارك.. لقد حصلت على تقنية صفحات العرض الخاصة بك.. ولا ينقص هذه التقنية إلا شيء واحد.. فمن المستحيل وضع كل أرقام الصفحات في لافتات الربط عندما يكون عدد الصفحات كبيرا (١٠٠ صفحة مثلا).. في هذه الحالة عليك أن تعرض أول عشرة أرقام فقط، مع وضع لافتة مكتوب عليها "التالي"، وعند الضغط عليها تعرض عشرة أرقام تالية، مع عرض لافتة في البداية اسمها "السابق"، عند الضغط عليها تعرض ١٠ أرقام سابقة.

دعنا نرى كيف نفعل هذا:

- أضف إلى النموذج لافتة رابط وامنحها الاسم LnkPrv واجعلها تعرض النص "السابق".
- أضف لافتة رابط أخرى اسمها LnkNext تعرض النص "التالي".
- عرف ثابتا على مستوى النموذج اسمه MaxLinks لنتحكم به في أقصى عدد يمكن عرضه من الروابط.
- عرف المتغيرات التالية على مستوى النموذج:

١- PageLinks، وهو قائمة مخصصة للتعامل مع لافتات الروابط List(Of LinkLabel)، لنضع فيها مراجع إلى لافتات الروابط التي نعرضها على النموذج.. هذا سيسهل علينا التحكم في هذه الروابط.

٢- CurLink، وهو يحمل مرجعا إلى لافتة الرابط المضغوطة حاليا، (التي تظهر الصفحة المناظرة لها في جدول العرض).. هذا سيفيدنا في التحرك إلى الأمام أو الخلف عند ضغط "التالي" أو "السابق".

٣- FirstPageNo، وهو متغير يحمل رقم أول صفحة يظهر حاليا في لافتات الروابط.. في البداية تكون قيمة هذا المتغير ١، ويمكن أن يتغير إذا عرضنا مجموعة أخرى من الروابط بسبب ضغط "التالي".

٤- PagesNo، وهو متغير يحمل عدد الصفحات الكلي الذي نتعامل معه.

- في حدث تحميل النموذج، سنعدل حلقة التكرار التي تضيف اللافتات، بحيث يكون أقصى عدد نضيفه هو MaxLinks.. يتم هذا كالتالي:

```
For I = 2 To Math.Min(PagesNo, MaxLinks)

Next
```

- سيظل كود حلقة التكرار كما كان في المشروع السابق، ما عدا زيادة سطر واحد، يضيف كل لافتة ننشئها إلى المجموعة PageLinks.. لا تنس أيضا إضافة اللافتة الأولى LnkPage1 إلى القائمة:

```
PageLinks.Add(LnkPage1)
For I = 2 To Math.Min(PagesNo, MaxLinks)
    ' نفس الكود القديم
    PageLinks.Add(PrvLnk)
Next
```

- استخدم الحدث LinkClicked لتعطيل الرابط "السابق" إذا كان رقم الرابط المضغوط ١، وتعطيل الرابط "التالي" إذا كان رقم الرابط المضغوط يساوي عدد الصفحات:

```
Dim I = CInt(Lnk.Text)
LnkPrev.Enabled = (I > 1)
LnkNext.Enabled = (I < PagesNo)
```

- سيكون من المفيد أيضا أن نميز الرابط المضغوط حاليا عن باقي الروابط، وذلك بتعطيله (فلا فائدة من ضغطه ثانية) وجعل خطه سميكا.. وعليك إعادة الرابط السابق إلى وضعه الطبيعي قبل تغيير حالة الرابط المضغوط حاليا.. أنسب مكان لفعل هذا هو الحدث LinkClicked أيضا:

```
' إعادة الرابط السابق إلى وضعه الأصلي
CurLink.Enabled = True
CurLink.Font = New Font(CurLink.Font,
          FontStyle.Regular)
' تمييز الرابط الحالي
Lnk.Enabled = False
Lnk.Font = New Font(Lnk.Font, FontStyle.Bold)
CurLink = Lnk
```

- في حدث ضغط "التالي" سنضغط الرابط الذي يزيد رقمه عن الرابط الحالي بواحد، إن كان معروضا على الشاشة:

```
Dim I = CurLink.Text
LnkPage1_LinkClicked(PageLinks(I - FirstPageNo + 1),
     New LinkLabelLinkClickedEventArgs(Nothing))
```

أما إذا كان الرابط السابق هو آخر رابط معروض على الشاشة، فيجب أن نعرض مجموعة تالية من الروابط.. لفعل هذا لا نحتاج إلى حذف الروابط الحالية وإنشاء روابط جديدة، فبإمكاننا أن نغير الأرقام المعروضة على اللافتات ببساطة، وذلك بجمع القيمة MaxLinks على كل منها.. لاحظ أن ناتج الجمع قد يتجاوز إجمالي عدد الصفحات في بعض الحالات، لهذا علينا إخفاء اللافتة التي يحدث لها هذا.. ولا تنسَ تغيير قيمة المتغير FirstPageNo، لتشير إلى رقم أول رابط في المجموعة الجديدة من الروابط.. هذا هو كود الحدث كاملا:

```
Dim I = CurLink.Text
If I = MaxLinks Then
   For Each Lnk In PageLinks
         Lnk.Text += MaxLinks
         If Lnk.Text > PagesNo Then Lnk.Visible = False
   Next
   FirstPageNo += MaxLinks
End If
LnkPage1_LinkClicked(PageLinks(I - FirstPageNo + 1),
     New LinkLabelLinkClickedEventArgs(Nothing))
```

- أخيرا، لم يتبق لنا إلا حدث ضغط "السابق".. هذا الكود مشابه لكود حدث ضغط "التالي"، مع عكس عمليات الجمع إلى طرح، وإظهار اللافتات المختفية.. كما أن شرط عرض المجموعة السابقة من اللافتات، هو أن يكون الرباط المضغوط حاليا أول رابط معروض على الشاشة.. هذا هو الكود:

```
Dim I = CurLink.Text
If I = FirstPageNo Then
  For Each Lnk In PageLinks
        Lnk.Text -= MaxLinks
        Lnk.Visible = True
  Next
  FirstPageNo -= MaxLinks
End If
LnkPage1_LinkClicked(PageLinks(I - FirstPageNo - 1),
     New LinkLabelLinkClickedEventArgs(Nothing))
```

الآن، يمكنك تشغيل البرنامج والاستمتاع بتجربته.. ستجد أن لديك بالفعل جدول عرض مقسم إلى صفحات، يعمل بكفاءة تامة، وقدرات كاملة!

-١٧-

# شبكة البيانات DataGrid

تعمل هذه الأداة كجدول يعرض أعمدة وصفوف، ويمكن ربطه بمصادر البيانات، بنفس الطريقة التي رأيناها في الأداة DataGridView.

ورغم أن الأداة DataGridView تمتلك قدرات أكثر من هذه الأداة، وتمنحك تحكما كاملا في كل أجزائها، وتتيح لك إنشاء أنواع جديدة من الخانات على حسب احتياجك، إلا أن شبكة البيانات تمتاز بخاصيتين لا توجدان في جدول العرض:

١- وجود عدة طرازات Styles جاهزة تمكنك من اختيار شكل الجدول مباشرة في وقت التصميم.

٢- قدرة شبكة البيانات على عرض الجداول المترابطة معا بطريقة تشبه Access، حيث يعرض كل صف في الجدول الرئيسي العلامة +، وعند ضغطها يتم عرض الصفوف الفرعية التابعة له من الجدول الثانوي.. ونظرا لأن الأداة DataGridView لا تملك مثل هذه الخاصية، فإن عليك أن تستخدم أداتين منها لعرض الجدول الأصلي والجدول الفرعي، بإضافة عمود أزرار إلى الجدول الرئيسي، وعند ضغط أي زر، يتم عرض نموذج جديد به جدول عرض يحتوي على السجلات الفرعية، كما فعلنا في المشروع DataGridViewAuthorBooks.. أو يمكنك عرض الجدول الفرعي في نفس النموذج أسفل الجدول الرئيسي، وقد رأينا كيف نفعل هذا في المشروع DataGridViewMasterDetails.

ونظرا لأهمية الخاصية الثانية، والتي قد تدفعك إلى استخدام شبكة البيانات في بعض مشاريعك على سبيل التسهيل، فقد رأيت أنه من الأفضل أن نتعرف على الأداة DataGrid، وهي على كل حال، ليست بضخامة الأداة DataGridView.

وعليك إضافة هذه الأداة أولا إلى صندوق الأدوات، بالضغط بزر الفأرة الأيمن في أي موضع تحت الشرط Data، وضغط الأمر Choose Items، ومن ثم اختيار العنصر DataGrid من قائمة الأدوات، مع التأكد أن العمود الثاني في القائمة يشير إلى أن العنصر الذي اخترته ينتمي إلى النطاق System.Windows.Forms، لأن هناك أداة شبكة بيانات أخرى خاصة بتطبيقات الويب.

## واجهة خدمة التحرير
## IdataGridEditingService Interface

تمنح هذه الواجهة مستخدم شبكة البيانات القدرة على تحرير خاناته، وهي تملك الوسيلتين التاليتين:

**بدء التحرير BeginEdit:**

تبدأ تحرير خانة في شبكة البيانات، وهي تستقبل المعاملين التاليين:

- كائن طراز العمود DataGridColumnStyle الذي يمثل العمود الذي توجد به الخانة المراد تحريرها.
- رقم الصف الذي توجد به الخانة المراد تحريرها.

وتعيد هذه الوسيلة True إذا نجحت في بدء التحرير.

**إنهاء التحرير EndEdit:**

تنهي تحرير خانة في شبكة البيانات، وهي تستقبل المعاملين التاليين:

- كائن طراز العمود DataGridColumnStyle الذي يمثل العمود الذي توجد به الخانة المراد تحريرها.
- رقم الصف الذي توجد به الخانة المراد تحريرها.
- معامل منطقي إذا جعلته True، فسيتم إلغاء القيمة الجديدة التي تم تحريرها، والعودة إلى القيمة الأصلية للخانة.. أما إذا جعلتها False، فسيتم حفظ القيمة الجديدة في الخانة.

وتعيد هذه الوسيلة True إذا نجحت في إنهاء التحرير، وتعيد False إذا فشلت في حفظ قيمة الخانة، وفي هذه الحالة تظل الخانة في وضع التحرير.

## فئة طراز شبكة البيانات DataGridTableStyle Class

هذه الفئة ترث الفئة Component، وتمثل الواجهة IdataGridEditingService، وهي تتيح لك التحكم في شكل الجدول الذي ترسمه الأداة DataGrid عند عرض البيانات.

ولحدث إنشاء هذه الفئة الصيغ التالية:

١- الصيغة الأولى بدون معاملات.

٢- الصيغة الثانية تستقبل معاملا منطقيا، إذا جعلت قيمته True، فسيشير هذا إلى أن طراز الجدول الحالي هو الطراز الافتراضي.

٣- الصيغة الثالثة تستقبل مدير التداول CurrencyManager الذي يتحكم في الارتباط بمصدر البيانات، ليستخدمه طراز الجدول في إنشاء الأعمدة وعرض البيانات فيها.

### شبكة البيانات DataGrid:

تقرأ أو تغير كائن شبكة البيانات DataGrid الذي يستخدم طراز الجدول.

### اسم الخريطة MappingName:

ضع في هذه الخاصية اسم عنصر البيانات الذي يعرضه طراز الجدول الحالي، وفي الغالب سيكون اسم أحد جداول مجموعة البيانات، مثل "Authors".. هذا يتيح لك استخدام أكثر من طراز جدول في شبكة البيانات، كل منها يعرض جدولا من جداول قاعدة البيانات بشكل وألوان وتنسيق خاصة به.

### طرازات أعمدة الجدول GridColumnStyles:

تعيد مجموعة طرازات أعمدة الجدول GridColumnStylesCollection، وهي ترث الفئة BaseCollection، كما أنها تمثل واجهة القائمة IList.. وتحتوي هذه المجموعة على عناصر من النوع DataGridColumnStyle، كل منها يمثل أحد الأعمدة المرسومة في طراز الجدول الحالي.. وسنتعرف على الفئة DataGridColumnStyle لاحقا.

### عناوين الأعمدة مرئية ColumnHeadersVisible:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتم عرض صف رؤوس الأعمدة.

**عناوين الصفوف مرئية RowHeadersVisible:**
إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيتم عرض عمود رؤوس الصفوف.

**العرض المفضل للعمود PreferredColumnWidth:**
تقرأ أو تغير العرض المبدئي الذي ستأخذه الأعمدة عند إنشائها في طـراز الجدول الحالي.

**الارتفاع المفصل للصفوف PreferredRowHeight:**
تقرأ أو تغير الارتفاع المبدئي الذي ستأخذه الصفوف عند إنشائها في طراز الجدول الحالي.

**عرض عناوين الصفوف RowHeaderWidth:**
تقرأ أو تغير عرض عمود رؤوس الصفوف.

**للقراءة فقط ReadOnly:**
إذا جعلت قيمة هذه الخاصية True، فلن يتمكن المستخدم من تغيير قيمة أي خانة في طراز الجدول الحالي.

**السماح بالترتيب AllowSorting:**
إذا جعلت قيمة هذه الخاصية True، فسيتمكن المستخدم من ترتيب صفوف الجدول تبعا للعمود الذي يضغط خانة عنوانه.

**لون الخلفية BackColor:**
تقرأ أو تغير لون خلفية الصفوف الزوجية في شبكة البيانات.

**لون الخلفية التبادلي AlternatingBackColor:**
تقرأ أو تغير لون خلفية الصفوف الفردية في شبكة البيانات.

**لون النص ForeColor:**
تقرأ أو تغير اللون المستخدم في كتابة نصوص الخانات.

**لون خلفية التحديد SelectionBackColor:**
تقرأ أو تغير لون خلفية الخانات المحددة.

**لون نص التحديد SelectionForeColor:**
تقرأ أو تغير لون نصوص خلفية الخانات المحددة.

**لون خطوط الشبكة GridLineColor:**
تقرأ أو تغير لون الخطوط الفاصلة بين الصفوف والأعمدة.

**طراز خطوط الشبكة GridLineStyle:**
تتحكم في شكل الخطوط الفاصلة بين الصفوف والأعمدة، وهي تأخذ إحدى قيمتي المرقم DataGridLineStyle التاليتين:

| | |
|---|---|
| None | لا يتم رسم خطوط الشبكة. |
| Solid | يتم رسم خطوط الشبكة. |

**لون خلفية العناوين HeaderBackColor:**
تقرأ أو تغير لون خلفية خانات عناوين الأعمدة والصفوف.

**لون نصوص العناوين HeaderForeColor:**
تقرأ أو تغير لون نصوص خانات عناوين الأعمدة والصفوف.

**خط العناوين HeaderFont:**
تقرأ أو تغير خط الكتابة المستخدم في خانات عناوين الأعمدة والصفوف.

**لون الروابط LinkColor:**
تقرأ أو تغير لون نصوص الوصلات Links الموجودة في خانات الجدول.

**لون التحليق فوق الروابط LinkHoverColor:**
تقرأ أو تغير لون نصوص الوصلات Links الموجودة في خانات الجدول، عند التحليق فوقها بالفأرة.

وتمتلك هذه الفئة عدة وسائل، لكنها غير هامة، فكلها تبدأ بالكلمة Reset متبوعة باسم إحدى الخصائص، ومهمتها إعادة قيمة تلك الخاصية إلى قيمتها الافتراضية، مثل ResetBackColor.
كما تمتلك هذه الفئة عدة أحداث، لكنها كلها تنطلق عند تغير قيمة إحدى الخصائص، مثل الحدث RowHeaderWidthChanged الذي ينطلق عندما يتغير عرض عمود رؤوس الصفوف.

## واجهة التنبيه بتحرير عمود شبكة البيانات IDataGridColumnStyleEditingNotificationService Interface

تنبه هذه الواجهة عمود شبكة البيانات بأن هناك أداة تحرير تستخدم حاليا مع إحدى خاناته، وهي تمتلك الوسيلة التالية:

**بدء تحرير العمود ColumnStartedEditing:**
تخبر شبكة البيانات بأن المستخدم بدأ تحرير إحدى الخانات، ولها معامل واحد من النوع Control، يستقبل أداة التحرير المستخدمة في الخانة.

## فئة طراز العمود DataGridColumnStyle

هذه الفئة أساسية مجردة Abstract Base Class تجب وراثتها، وهي ترث الفئة Component، كما أنها تمثل الواجهة IDataGridColumnStyleEditingNotificationService.. وتعمل الفئات المشتقة من هذه الفئة كأعمدة في شبكة البيانات، كما أنها تتحكم في شكل وتنسيق خانات هذه الأعمدة.

وتمتلك هذه الفئة الخصائص التالية:

**طراز شبكة البيانات DataGridTableStyle:**
تعيد كائن طراز الجدول DataGridTableStyle، الذي يحتوي على كائن طراز العمود الحالي.

**المحاذاة Alignment:**
تحدد محاذاة النص في خانات العمود، وهي تأخذ إحدى قيم المرقم HorizontalAlignment التالية: Left – Right – Center.

**عنوان العمود HeaderText:**
تقرأ أو تغير النص المعروض في خانة عنوان العمود.

**اسم الخريطة MappingName:**
تحدد اسم عنصر البيانات Data Member الذي يعرضه طراز العمـود.. وفي الغالب تحتوي هذه الخاصية على اسم أحد أعمدة مجموعة البيانات.

**النص المنعدم NullText:**
ضع في هذه الخاصية النص الذي ستعرضه خانات العمود إذا كانت فارغة.

**للقراءة فقط ReadOnly:**
إذا جعلت قيمة هذه الخاصية True، فلن يستطيع المستخدم تغيير قيمة أيـة خانة في العمود الحالي، والقيمة الافتراضية هي False.

**العرض Width:**
تقرأ أو تغير عرض العمود الحالي.

**واصف الخاصية PropertyDescriptor:**
تقرأ أو تغير كائن واصف الخاصية PropertyDescriptor الذي يحـوي على سمات العمود.. هذا يمكنك من تحديد نوع البيانات التي يقبلها العمود.

وتمتلك هذه الفئة الوسيلة التالية:

**تصفير عنوان العمود ResetHeaderText:**
تعيد الخاصية HeaderText إلى قيمتها الافتراضية، وهي نص فارغ "".

كما تمتلك هذه الفئة مجموعة من الأحداث، أهمها الحدث التالي:

**العرض تغير WidthChanged:**
ينطلق إذا تغير عرض العمود.

## فئة عمود النصوص DataGridTextBoxColumn Class

هذه الفئة ترث الفئة DataGridColumnStyle، وهي تمثل عمودا في شبكة البيانات تعرض خاناته نصوصا.

ولحدث إنشاء هذه الفئة الصيغ التالية:

١- الصيغة الأولى بدون معاملات.

٢- الصيغة الثانية تستقبل واصف الخاصية PropertyDescriptor المستخدم مع العمود الحالي.

٣- الصيغة الثالثة تزيد على الصيغة السابقة بمعامل منطقي، إذا جعلته True فسيتم اعتبار العمود الحالي عمودا افتراضيا Default Column.

٤- الصيغة الرابعة تزيد على الصيغة الثانية بمعامل نصي، يستقبل الصيغة المستخدمة في تنسيق النصوص في خانات العمود.

٥- الصيغة الخامسة تزيد على الصيغة السابقة بمعامل منطقي، إذا جعلته True فسيتم اعتبار العمود الحالي عمودا افتراضيا Default Column.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

### التنسيق Format:

ضع في هذه الخاصية نصا يحمل الصيغة المستخدمة لتنسيق خانات العمود.

### معلومات التنسيق FormatInfo:

هذه الخاصية من النوع IFormatProvider، وهي تستقبل كائن معلومات الثقافة CultureInfo، الذي يحوي معلومات عن اللغة والمنطقة التي ستستخدم قواعدهما في تنسيق الأرقام والتواريخ والنصوص.. وفي الوضع الافتراضي، تستخدم هذه الخاصية الثقافة المحلية الخاصة بجهاز المستخدم.

### مربع النص TextBox:

تعيد كائن مربع النص TextBox، الذي يستضيفه العمود الحالي ويعرضه في الخانة التي يقوم المستخدم بتحريرها.

## فئة العمود المنطقي DataGridBoolColumn Class

هذه الفئة ترث الفئة DataGridColumnStyle، وهي تمثل عمودا في شبكة البيانات تعرض خاناته مربع اختيار CheckBox، ليمثل القيم المنطقية Boolean.
ولحدث إنشاء هذه الفئة الصيغ التالية:

١- الصيغة الأولى بدون معاملات.
٢- الصيغة الثانية تستقبل واصف الخاصية PropertyDescriptor المستخدم مع العمود الحالي.
٣- الصيغة الثالثة تزيد على الصيغة السابقة بمعامل منطقي، إذا جعلته True فسيتم اعتبار العمود الحالي عمودا افتراضيا Default Column.

وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الفئة الخصائص التالية:

### السماح بالعدم AllowNull:

إذا جعلت قيمة هذه الخاصية False، فسيكون لمربع الاختيار حالتان فقط: Checked و Unchecked.. والقيمة الافتراضية لهذه الخاصية هي True، وفي هذه الحالة سيكون لمربع الاختيار الحالة الثالثة غير المحددة Indeterminate إذا كانت قيمة الخانة DbNull.

### القيمة المنعدمة NullValue:

ضع في هذه الخاصية القيمة التي تعتبر عدما، والتي ستجعل مربع الاختيار في الحالة غير المحددة.. يمكنك مثلا استخدام الرقم -١ كمناظر للقيمة غير المحددة.

### القيمة الخاطئة FalseValue:

ضع في هذه الخاصية القيمة التي تعتبر False، والتي ستجعل مربع الاختيار في حالة عدم الاختيار Unchecked.. يمكنك مثلا استخدام الرقم صفر كمناظر للقيمة الخاطئة.

### القيمة الصحيحة TrueValue:

ضع في هذه الخاصية القيمة التي تعتبر True، والتي ستجعل مربع الاختيار في حالة الاختيار Checked.. يمكنك مثلا استخدام الرقم ١ كمناظر للقيمة الخاطئة.

## سجل خانة الشبكة

## DataGridCell Structure

يعمل هذا السجل كخانة في شبكة البيانات، ولحدث إنشائه معاملان:

- رقم الصف الذي توجد به الخانة.
- رقم العمود الذي توجد به الخانة.

ويمتلك هذا السجل الخاصيتين التاليتين:

### رقم الصف RowNumber:

تقرأ أو تغير رقم الصف الذي توجد به الخانة.

### رقم العمود ColumnNumber:

تقرأ أو تغير رقم العمود الذي توجد به الخانة.

## فئة شبكة البيانات DataGrid Class

هـذه الفئـة تـرث فئـة الأداة الأم Control، وتمثـل الواجهـة IdataGridEditingService، وهي تمتلك الخصائص التالية:

### مصدر البيانات DataSource:

ضع في هذه الخاصية الكائن الذي يعمل كمصدر للبيانات التـي تعرضـها شبكة البيانات، مثـل مجموعـة البيانـات DataSet أو جـدول البيانـات DataTable أو عرض البيانات DataView أو أية مجموعة تحتوي على كائنات بها خصائص عامة.

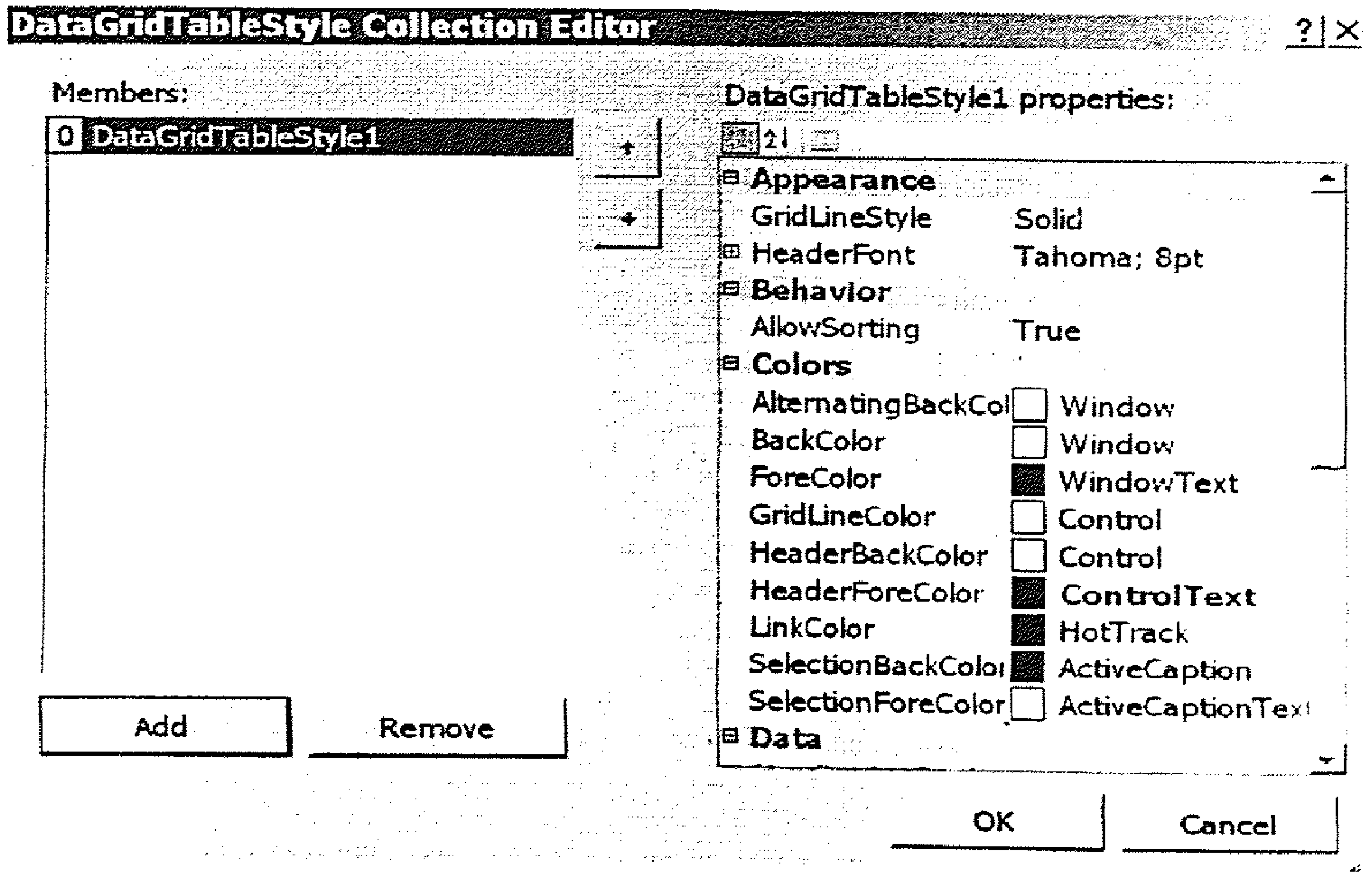
### عنصر البيانات DataMember:

ضع في هذه الخاصية اسم عنصر البيانات المراد عرضه، مثل اسم جـدول المؤلفين "Authors".

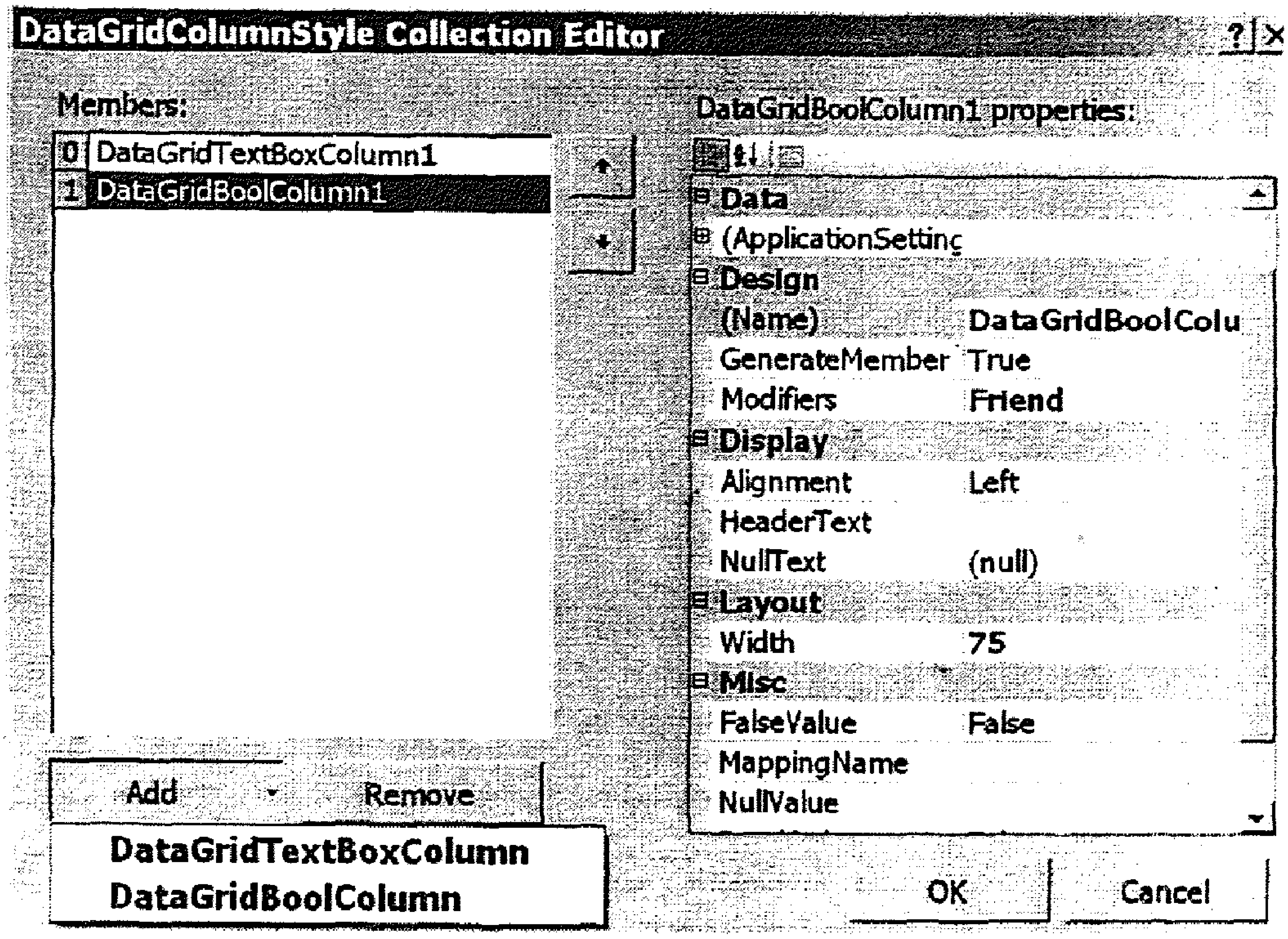
### طرازات الجداول TableStyles:

تعيد مجموعة طرازات الجداول GridTableStylesCollection، وهـي ترث الفئة BaseCollection وتمثل واجهة لقائمة IList، وكل عنصر من عناصرها من النوع GridTableStyles.

ويمكنك إضافة العناصر إلى هذه المجموعة بشكل مرئي في وقت التصميم، وذلك باستخدام نافذة الخصائص، حيث يؤدي ضغط الزر الموجود في خانة هذه الخاصية إلى عرض النافذة الموضحة في الصورة:

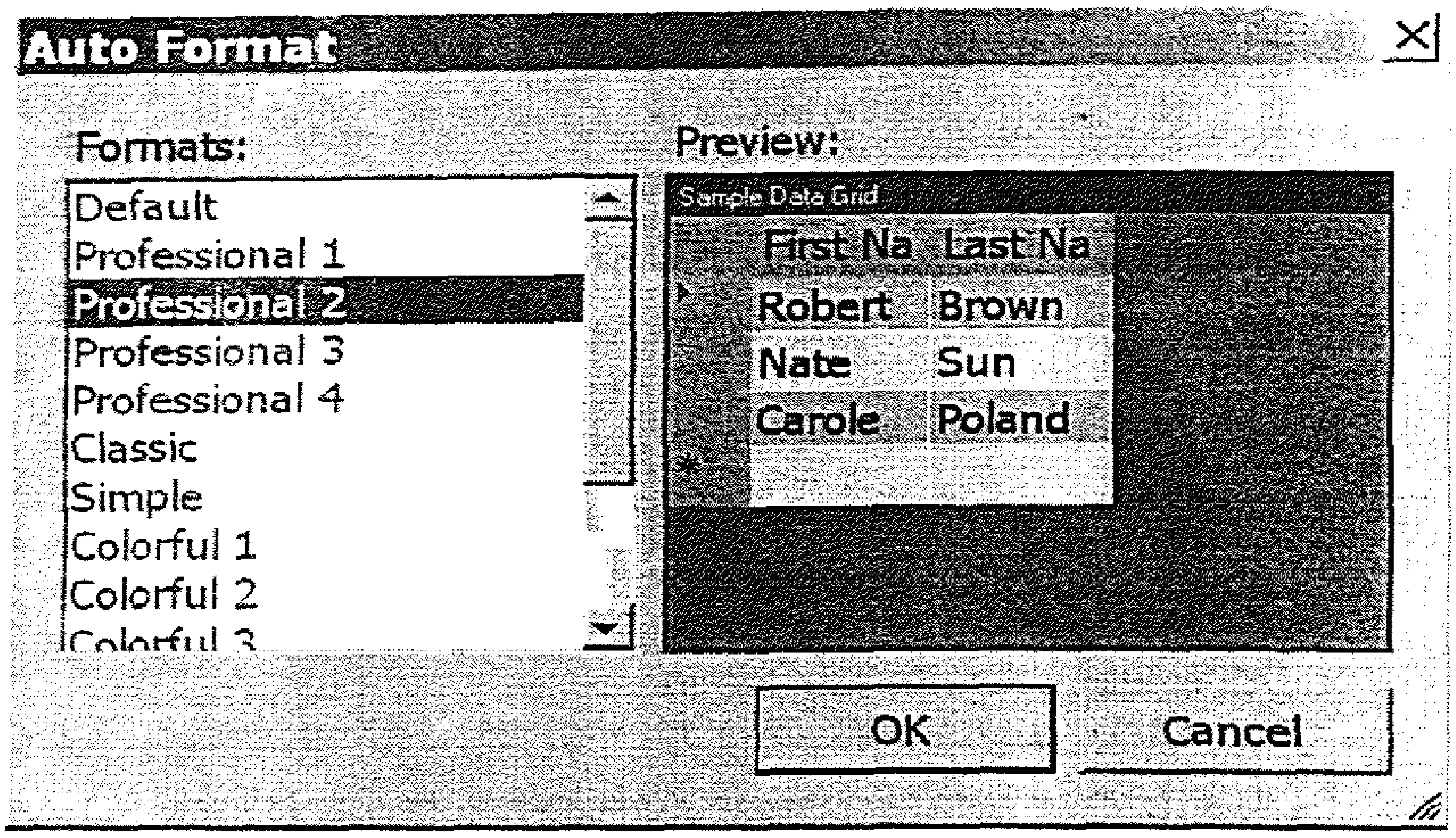


في هذه النافذة، يمكنك ضغط الزر Add لإضافة طراز جدول جديد، واستخدام الخصائص المعروضة في النصف الأيمن من النافذة للتحكم في خصائص هذا الطراز، وربطه بجدول البيانات باستخدام الخاصية MappingName.. كما يمكنك استخدام الخاصيّة GridColumnStyles للتحكم في طراز أعمدة الجدول.. لفعل هذا اضغط زرّ الانتقال المجاور لهذه الخاصيّة، لتظهر لك نافذة محرر مجموعة طرازات الأعمدة:




في هذه النافذة يمكنك إضافة طراز عمود جديد بضغط الزر Add.. لاحظ وجود سهم في جانب الزر Add، ولو ضغطته بالفأرة، فستظهر قائمة موضعية، تتيح لك اختيار نوع العمود الذي تريد إضافة طراز له، سواء كان عمود نصوص DataGridTextBoxColumn، أو عمودا منطقيا DataGridBoolColumn.. ويمكنك تغيير خصائص طراز العمود من النصف الأيمن للنافذة.. ولا تنسَ أن تضع في الخاصيّة MappingName اسم العمود الأصلي الذي يعرض طراز العمود بياناته.

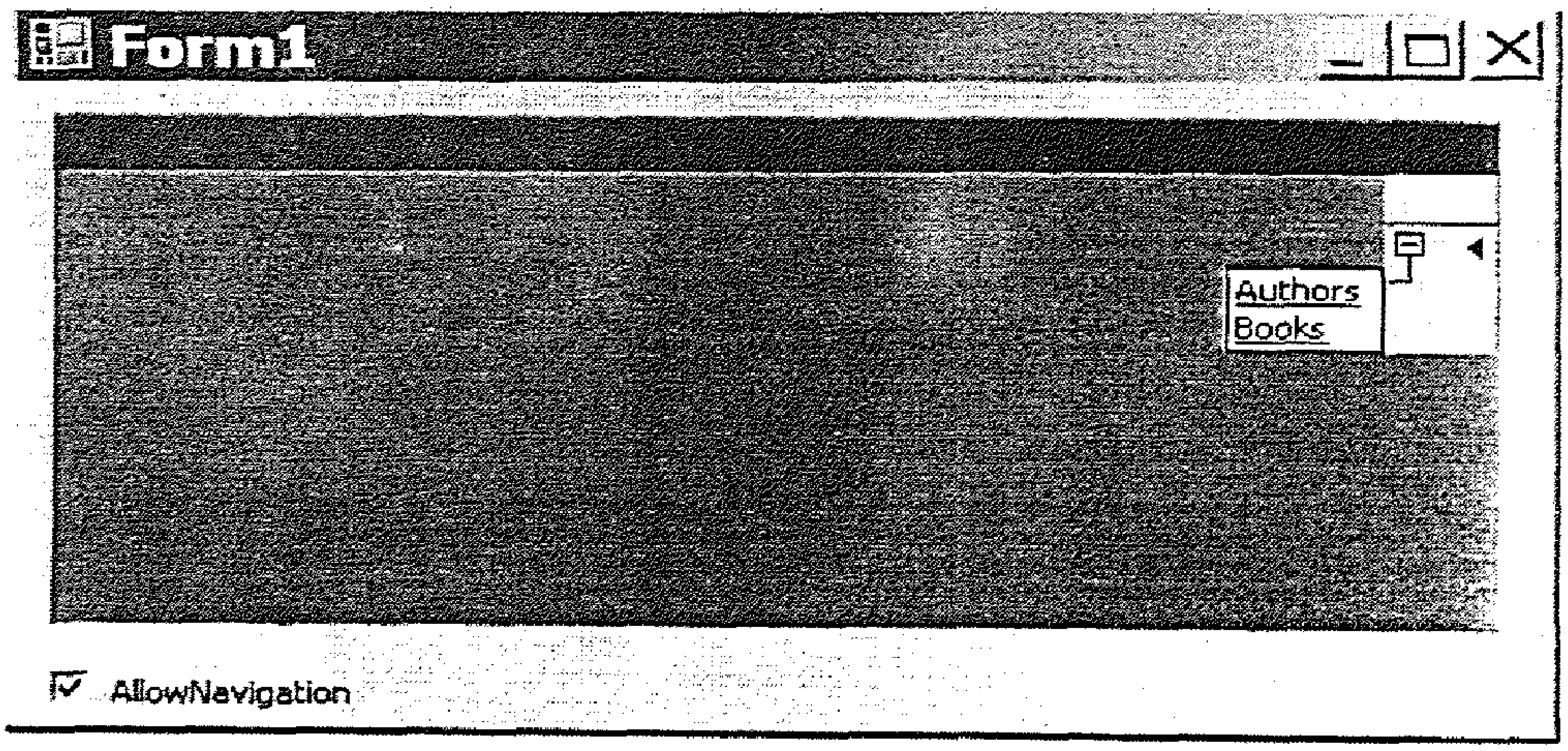
كما يمكنك أن تحصل على تنسيقات جاهزة للجدول، وذلك بضغط شبكة البيانات بزرّ الفأرة الأيمن في وقت التصميم، واختيار الأمر AutoFormat من القائمة الموضعيّة، حيث ستظهر لك نافذة تحتوي على قائمة بأسماء التنسيقات المتاحة، مع عرض نموذج لتأثير كلّ منها على جدول المعاينة.




### السماح بالتصفح AllowNavigation:

إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فستعرض شبكة البيانات الوصلات التي تتيح لك استعراض الجداول الفرعية.

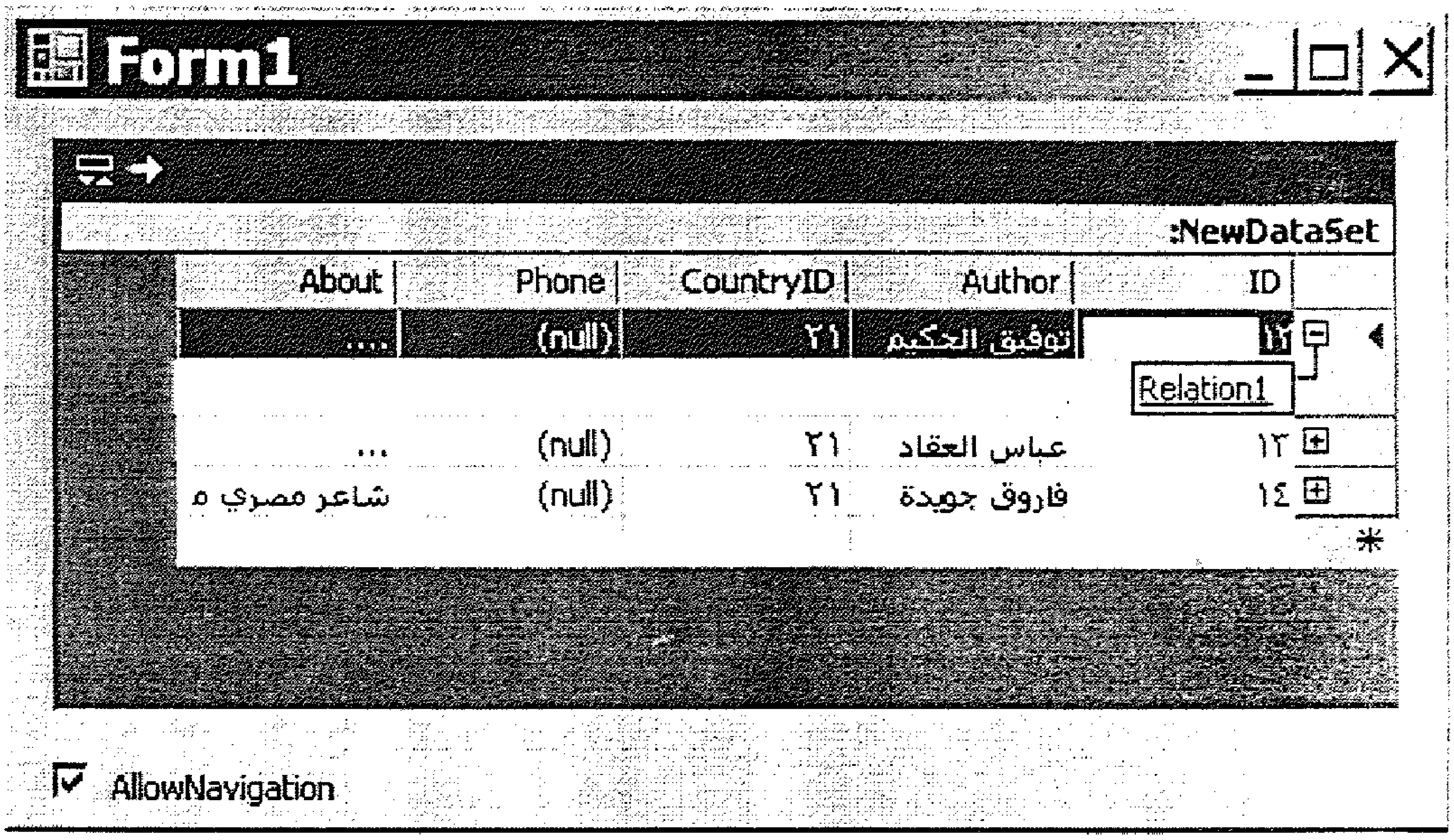
ويمكنك اختبار تأثير هذه الخاصية في التطبيق DataGridNavigation، فلو شغّلت هذا البرنامج فلن تظهر في الجدول أيّ بيانات.. وإنما ستظهر العلامة "+"، ولو ضغطتها فستعرض لك اسمي جدولي المؤلفين والكتب.



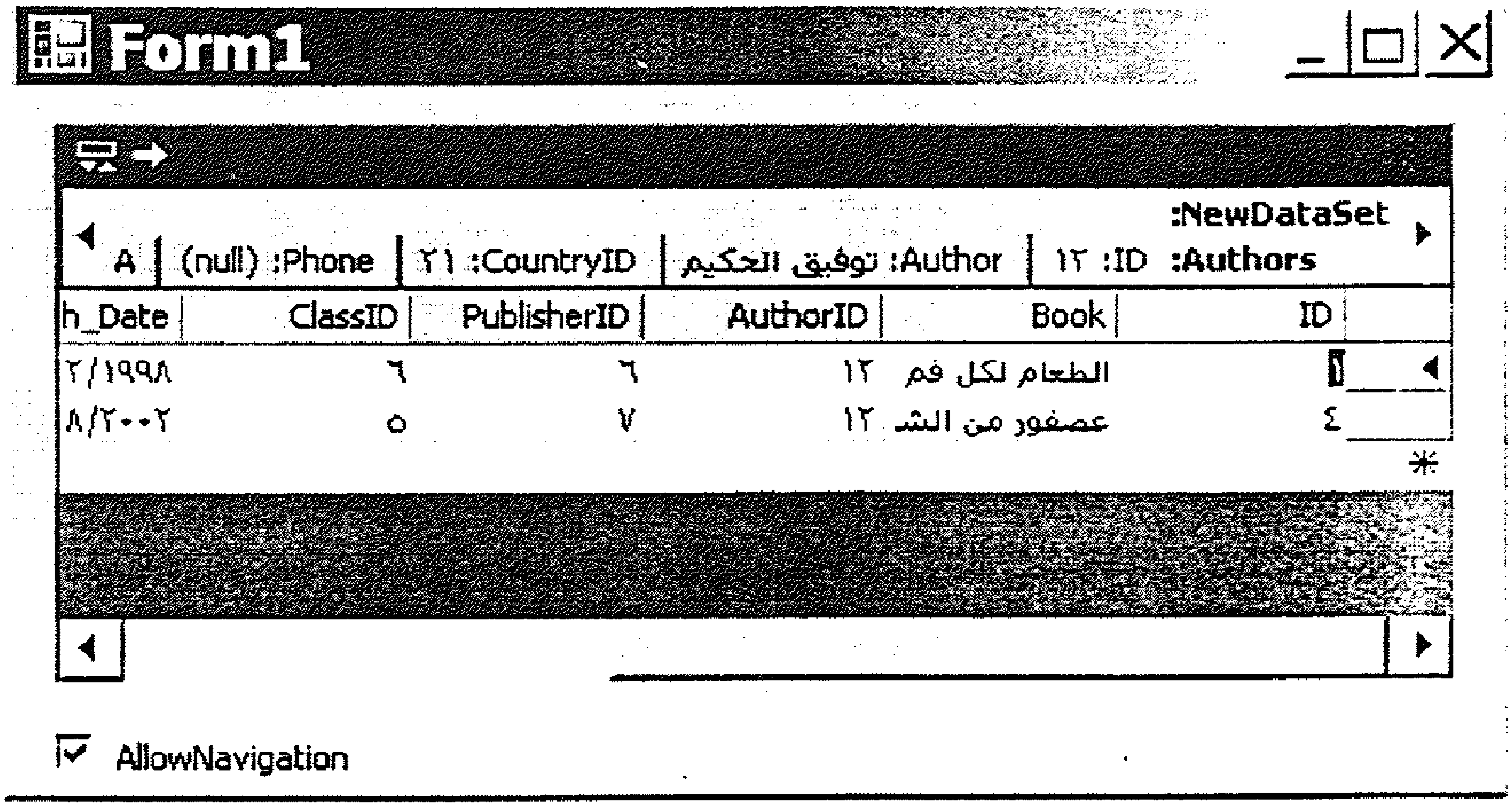

ولو أزلت علامة الاختيار من مربع الاختيار AllowNavigation، فستختفي العلامة + واسما الجدولين، وسيظل جدول العرض فارغا!

أعد وضع علامة الاختيار، واضغط اسم جدول المؤلفين Authors.. سيمتلئ الجدول بسجلات جدول المؤلفين، وفي الهامش العلوي لشبكة البيانات سيظهر اسم مجموعة البيانات.. لهذا سيكون مفيدا أن تعطي

مجموعة البيانات اسما واضحا بدلا من الاسم الافتراضي NewDataSet.. وقد أسميناها في هذا المشروع "قاعدة بيانات الكتب".
وأمام كلّ سجلّ من سجلات المؤلفين، سترى العلامة "+".. ولو ضغطت هذه العلامة، فسيظهر لك اسم العلاقة بين الجدولين، وسيكون مفيدا أيضا لو أعطيت العلاقة اسما واضحا للمستخدم بدلا من الاسم الافتراضي Relation1.. وقد أسميناها في هذا المشروع "كتب المؤلف":




وأيضا، لو أزلت علامة الاختيار من مرع الاختيار AllowNavigation، فستختفي العلامة + وستظهر سجلات جدول المؤلفين بمفردها.
اضغط اسم العلاقة بالفأرة.. سيتمّ عرض أسماء الكتب الخاصّة بهذا المؤلف، كما سيظهر سجل هذا المؤلف أعلى الجدول:



ولو أردت إخفاء سجل المؤلف المعروض أعلى الجدول، فاضغط العلامة الموجودة أعلى يسار الجدول، ولإعادة عرض هذا السجل، فاضغط نفس العلامة مرة أخرى.. ولو أردت العودة إلى جدول المؤلفين، فاضغط السهم الموجود أعلى شبكة البيانات.
لاحظ أن شبكة البيانات تعرض اسمي جدولي المؤلفين والكتب عند تشغيل البرنامج، لأننا ربطناه بمجموعة البيانات كلها، ولو كانت فيها جداول أكثر لظهرت اسماؤها كلها:

**DataGrid1.DataSource = Ds**

ولو أردت عرض جدول واحد فقط كجدول المؤلفين، فاستخدم الجملة التالية:

**DataGrid1.DataSource = Ds.Tables("Authors")**

في هذه الحالة ستعرض شبكة البيانات سجلات جدول المؤلفين مباشرة، مع ظهور العلامة + بجوار كل سجل من سجلاته، ليمكنك عرض السجلات الفرعية في جدول الكتب.. أما لو استخدمت الجملة التالية:

**DataGrid1.DataSource = Ds.Tables("Books")**

فستعرض شبكة البيانات سجلات جدول الكتب فقط، ولن تجد أية طريقة لعرض أية سجلات من جدول المؤلفين.

### لون الأرضية BackgroundColor:

تقرأ أو تغير لون أرضية شبكة البيانات (الجزء الذي لا توجد فيه خانات).

### العنوان مرئي CaptionVisible:

إذا جعلت قيمة هذه الخاصية False، فلن يظهر شريط العنوان أعلى شبكة البيانات.. والقيمة الافتراضية لهذه الخاصية هي True.

### نص العنوان CaptionText:

تقرأ أو تغير النص المعروض في شريط العنوان.

### لون خلفية العنوان CaptionBackColor:

تقرأ أو تغير لون خلفية شريط العنوان.

### لون نص العنوان CaptionForeColor:

تقرأ أو تغير لون النص المعروض في شريط العنوان.

**خط العنوان CaptionFont:**
تقرأ أو تغير الخط المستخدم لكتابة النص المعروض في شريط العنوان.

**رقم الصف الحالي CurrentRowIndex:**
تقرأ أو تغير رقم الصف المحدد حاليا في شبكة البيانات.

**الخانة الحالية CurrentCell:**
تعيد خانة شبكة البيانات DataGridCell المحددة حاليا.

**العنصر Item:**
تقرأ أو تغير قيمة خانة معينة في جدول العرض.. ولهذه الخاصية صيغتان:
١- الأولى تستقبل كائن الخانة DataGridCell التي تريد التعامل معها.
٢- الثانية تستقبل رقم الصف ورقم العمود اللذين توجد فيهما الخانة.

**أول عمود مرئي FirstVisibleColumn:**
تعيد رقم أول عمود ظاهر على الشاشة.

**عدد الأعمدة المرئية VisibleColumnCount:**
تعيد عدد الأعمدة الظاهرة حاليا على الشاشة.

**عدد الصفوف المرئية VisibleRowCount:**
تعيد عدد الصفوف الظاهرة حاليا على الشاشة.

**العرض المسطح FlatMode:**
إذا جعلت قيمة هذه الخاصية True، فستظهر شبكة البيانات بشكل مسطح.. والقيمة الافتراضية لهذه الخاصية هي False.

**الصفوف الرئيسية مرئية ParentRowsVisible:**
إذا جعلت قيمة هذه الخاصية True (وهي القيمة الافتراضية)، فسيظهر الصف الرئيسي أعلى شبكة العرض عند عرض الصفوف الفرعية.

**لون خلفية الصفوف الرئيسية ParentRowsBackColor:**
تقرأ أو تغير لون خلفية الصف الرئيسي.

**لون نص الصفوف الرئيسية ParentRowsForeColor:**
تقرأ أو تغير لون نص الصف الرئيسي.

**طراز لافتة الصفوف الرئيسية ParentRowsLabelStyle:**
تتحكم في عنوان لافتة الصف الرئيسي، وهي تأخذ إحدى قيم المرقم DataGridParentRowsLabelStyle التالية:

| | |
|---|---|
| None | لا تعرض اللافتة أي عنوان. |
| TableName | تعرض اللافتة اسم الجدول. |
| ColumnName | تعرض اللافتة اسم العمود الرئيسي (المفتاح الأساسي). |
| Both | تعرض اللافتة اسم الجدول واسم العمود الرئيسي. |

كما يمتلك جدول العرض الخصائص التالية:

- **عناوين الأعمدة مرئية ColumnHeadersVisible**
- **عناوين الصفوف مرئية RowHeadersVisible**
- **العرض المفضل للعمود PreferredColumnWidth**
- **الارتفاع المفصل للصفوف PreferredRowHeight**
- **عرض عناوين الصفوف RowHeaderWidth**
- **للقراءة فقط ReadOnly**
- **السماح بالترتيب AllowSorting**
- **لون الخلفية BackColor**
- **لون الخلفية المتبدل AlternatingBackColor**
- **لون النص ForeColor**
- **لون خلفية التحديد SelectionBackColor**
- **لون نص التحديد SelectionForeColor**
- **لون خطوط الشبكة GridLineColor**
- **طراز خطوط الشبكة GridLineStyle**

- لون خلفية العناوين **HeaderBackColor**
- لون نصوص العناوين **HeaderForeColor**
- خط العناوين **HeaderFont**
- لون الروابط **LinkColor**
- لون التحليق فوق الروابط **LinkHoverColor**

كما تلاحظ، فإن هذه الخصائص موجودة بنفس الاسم والوظيفة في كائن طراز الجدول DataGridTableStyle، لهذا لن نعيد شرحها هنا.. عليك فقط أن تعرف أن خصائص طراز الجدول يكون لها الأولوية على خصائص شبكة البيانات، ولا تؤثر هذه الأخيرة إلا على الجدول الذي تعرضه بدون إنشاء طراز جدول خاص به.

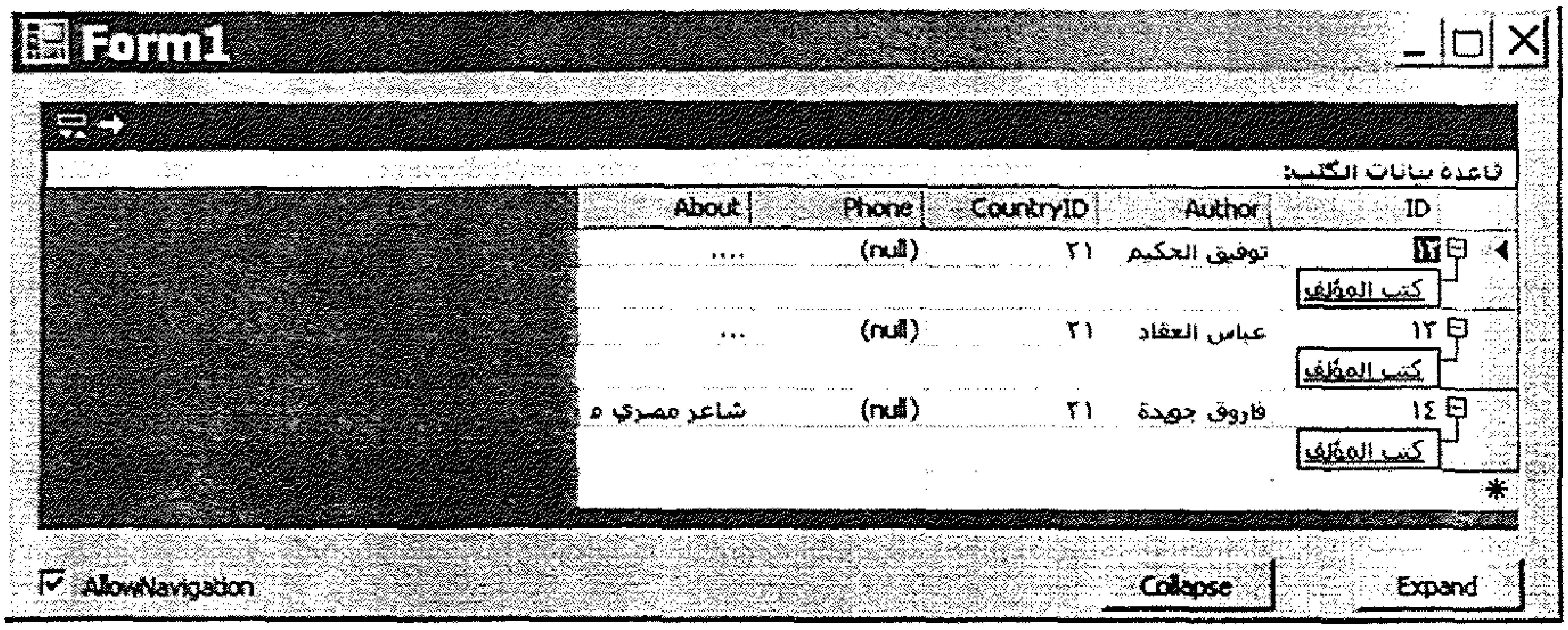
كما تمتلك شبكة العرض الوسائل التالية:

- إسدال **Expand**:

أرسل إلى هذه الوسيلة رقم الصف الذي تريد إسدال العلاقة التابعة له (كأنك ضغطت العلامة + المجاورة له).. فإذا لم تكن للصف المطلوب صفوف فرعية، فلن يحدث أي خطأ، ولن تفعل هذه الوسيلة شيئا.
ويمكنك إرسال الرقم ١- كمعامل لإسدال علاقات كل الصفوف:

**DataGrid1.Expand(-1)**

وستجد زرا اسمه Expand في المشروع DataGridNavigation، وعند الضغط عليه سيتم إسدال كل جداول مجموعة البيانات، أو إسدال اسم العلاقة لكل صف من صفوف جدول المؤلفين، كما في الصورة.
لاحظ أنك إذا أزلت علامة الاختيار من مربع الاختيار AllowNavigation، فستختفي العلامة + من جوار كل صف، لكن ضغط الزر Expand سيظل يسدل اسم العلاقة لكل صف، لكنها ستكون عاطلة عن العمل، ولن يؤدي ضغطها إلى عرض الصفوف الفرعية.




### هل هو مسدل IsExpanded:

تعيد True إذا كانت علاقة الصف الذي أرسلت إليها رقمه مسدلة.. لاحظ أن خطأ سيحدث إذا أرسلت إلى هذه الوسيلة الرقم -١.

### طي Collapse:

أرسل إلى هذه الوسيلة رقم الصف الذي تريد طي العلاقة التابعة له (كأنك ضغطت العلامة - المجاورة له).. فإذا لم تكن للصف المطلوب صفوف فرعية، فلن يحدث أي خطأ، ولن تفعل هذه الوسيلة شيئا.

ويمكنك إرسال الرقم -١ كمعامل لطي كل علاقات الصفوف:

**DataGrid1.Collapse (-1)**

وستجد زرا اسمه Collapse في المشروع DataGridNavigation، عند الضغط عليه يتم طي كل جداول مجموعة البيانات، أو طي اسم العلاقة لكل صف من صفوف جدول المؤلفين.

### معرفة حدود الخانة GetCellBounds:

تعيد كائن المستطيل Rectangle الذي يمثل موضع وأبعاد الخانة المرسلة كمعامل.. ولهذه الوسيلة صيغتان:

١- الأولى تستقبل كائن الخانة DataGridCell.

٢- والثانية تستقبل رقم الصف ورقم العمود اللذين توجد فيهما الخانة.

### معرفة حدود الخانة الحالية GetCurrentCellBounds:

تعيد كائن المستطيل Rectangle الذي يمثل موضع وأبعاد الخانة المحددة حاليا في شبكة البيانات.

- **هل هو محدد IsSelected:**
تعيد True إذا كان الصف الذي أرسلت إليها رقمه محددا.

- **تحديد Select:**
إضافة إلى الصيغة الموروثة من الفئة Control والتي تحدد شبكة البيانات نفسها (تنقل غليها المؤشر Focus)، توجد صيغة أخرى تحدد الصف الذي ترسل إليها رقمه كمعامل.

- **إلغاء التحديد UnSelect:**
تزيل تحديد الصف الذي ترسل إليها رقمه كمعامل.

- **تصفير التحديد ResetSelection:**
تزيل تحديد كل صفوف شبكة البيانات.

- **الانتقال إلى الخلف NavigateBack:**
تعرض الجدول الرئيسي الذي كان معروضا قبل الجدول الحالي في شبكة البيانات.. فمثلا: لو كانت كتب توفيق الحكيم معروضة حاليا، فستعيد هذه الوسيلة عرض جدول المؤلفين.. أما إذا لم يكن هناك جدول سابق، فلن يحدث خطأ في البرنامج، ولن تفعل هذه الوسيلة شيئا.. هذا معناه أنها تؤدي نفس وظيفة زر التراجع الموجود أعلى شبكة البيانات.. ويمكنك تجربة هذه الوسيلة بضغط الزر NavigateBack في المشروع DataGridNavigation.

- **الانتقال إلى NavigateTo:**
أرسل إلى هذه الوسيلة رقم الصف، واسم العلاقة، لعرض سجلاته الفرعية في شبكة البيانات، تماما كأن المستخدم ضغط العلاقة الخاصة بهذا الصف.. ولا تسبب هذه الوسيلة أي خطأ في البرنامج إذا لم يكن الصف المطلوب مرتبطا بالعلاقة المذكورة، ولكنها بدلا من هذا تقوم بالعودة إلى الجدول الرئيسي مجددا، كأنك استدعيت الوسيلة NavigateBack.. ويمكنك تجربة هذه الوسيلة بضغط الزر NavigateTo في المشروع DataGridNavigation.

- **تغيير ربط البيانات SetDataBinding:**
تربط شبكة العرض بمصدر البيانات، وهي تستقبل معاملين:
  - الكائن الذي يعمل كمصدر للبيانات Data Source.
  - اسم الجدول أو المجموعة التي تعمل كعنصر للبيانات.

### اختبار الضغط HitTest:

تعيد كائن معلومات اختبار الضغط HitTestInfo الذي يحوي معلومات عن النقطة المرسلة كمعامل إلى هذه الخاصية، سواء كانت في صورة كائن نقطة Point أو في صورة الإحداثيين الأفقي X والرأسي Y.

والفئة HitTestInfo معرفة داخل الفئة DataGrid، وهي تمتلك الخصائص التالية:

| | | |
|---|---|---|
| | Nowhere | تعيد كائن معلومات اختبار HitTestInfo، يشير إلى نقطة موجودة في منطقة فارغة من جدول العرض (ليست بها خانات عادية أو خانات عناوين). |
| | Column | تعيد رقم العمود الذي توجد فيه نقطة الاختبار. |
| | Row | تعيد رقم الصف الذي توجد فيه نقطة الاختبار. |
| | Type | تعيد إحدى قيم المرقم DataGrid.HitTestType، التي تخبرك بنوع المنطقة التي توجد بها نقطة الاختبار.. وهذه القيم هي:<br>- None: منطقة فارغة.<br>- Cell: خانة.<br>- ColumnHeader: رأس عمود.<br>- RowHeader: رأس صف.<br>- ColumnResize: الخط الرأسي الفاصل بين رأسي عمودين.<br>- RowResize: الخط الأفقي الفاصل بين رأسي صفين.<br>- Caption: شريط العنوان العلوي لشبكة البيانات.<br>- ParentRows: الصفوف الرئيسية المعروضة أعلى شبكة البيانات. |

كما تمتلك شبكة البيانات الأحداث التالية:

- **الخانة الحالية تغيرت CurrentCellChanged:**
ينطلق عندما تتغير الخانة المحددة حاليا في شبكة البيانات.

- **ضغط زر الرجوع BackButtonClick:**
ينطلق عندما يضغط المستخدم زر الرجوع إلى الخلف الموجود أعلى شبكة البيانات.

- **تصفح Navigate:**
ينطلق عندما تنتقل شبكة البيانات لعرض جدول آخر.. والمعامل الثاني e لهذا الحدث من النوع NavigateEventArgs، وهو يمتلك الخاصية Forward التي تعيد True إذا كان الانتقال إلى الأمام (إلى جدول فرعي).

- **ضغط زر عرض تفاصيل السجل الرئيسي ShowParentDetailsButtonClick:**
ينطلق عندما يضغط المستخدم زر عرض أو إخفاء تفاصيل السجل الرئيسي، المعروض أعلى شبكة البيانات.

- **انزلاق Scroll:**
ينطلق عندما يتحرك أحد المنزلقين الأفقي أو الرأسي في شبكة البيانات.

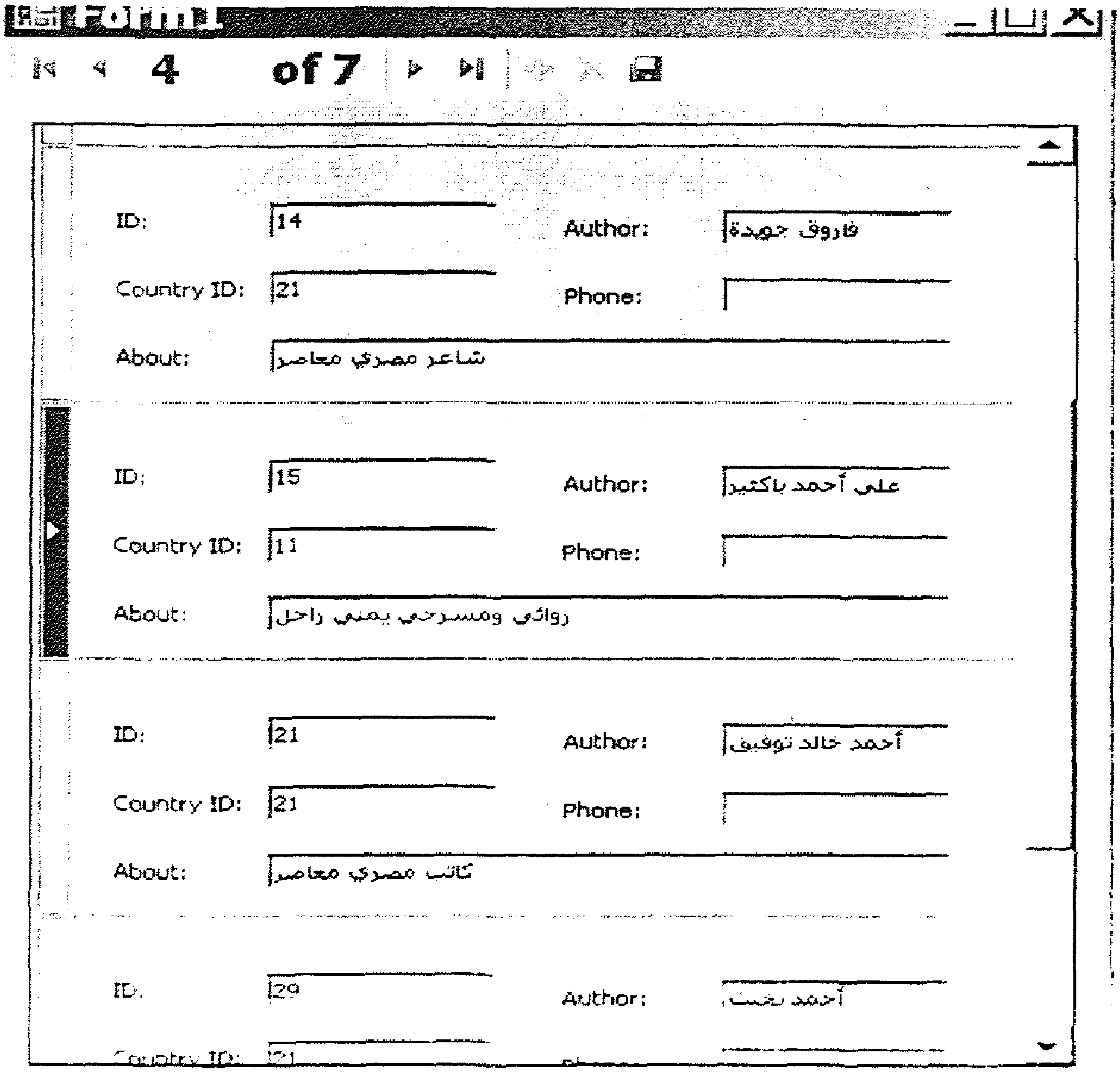
والآن، لعلك لاحظت مدى تواضع الأداة DataGrid بالنسبة إلى الأداة DataGridView، وإن كانت الأولى ما تزال صالحة للاستخدام في الحالات التي تريد فيها عرض بعض البيانات دون الحاجة إلى التحكم الكامل في الأعمدة والصفوف ومحتويات الخانات، فكما ترى، لا تقدم لك شبكة البيانات أية طريقة للتعامل مع الصفوف والأعمدة والخانات، إلا بمعرفة أرقامها، أو بتغيير طراز عرض الأعمدة، وهو ما يحد من قدرتك على برمجة هذه الأداة بشكل كبير، ولعله السبب الرئيسي الذي حدا ميكروسوفت إلى إنشاء جدول العرض!

-١٨-

# مُكرِّر البيانات

# Data Repeater

تمنحك هذه الأداة القدرة على عرض البيانات في صورة قائمة List من العناصر بالتنسيق الذي تريده.. وتختلف هذه الأداة عن القوائم التقليدية في أنها لا تعرض العنصر على شكل نص أو صورة، بل تتيح لك تصميم كل عنصر بأي عدد من الأدوات كما تريد، وبأي شكل تريد، كما تبين الصورة التالية:



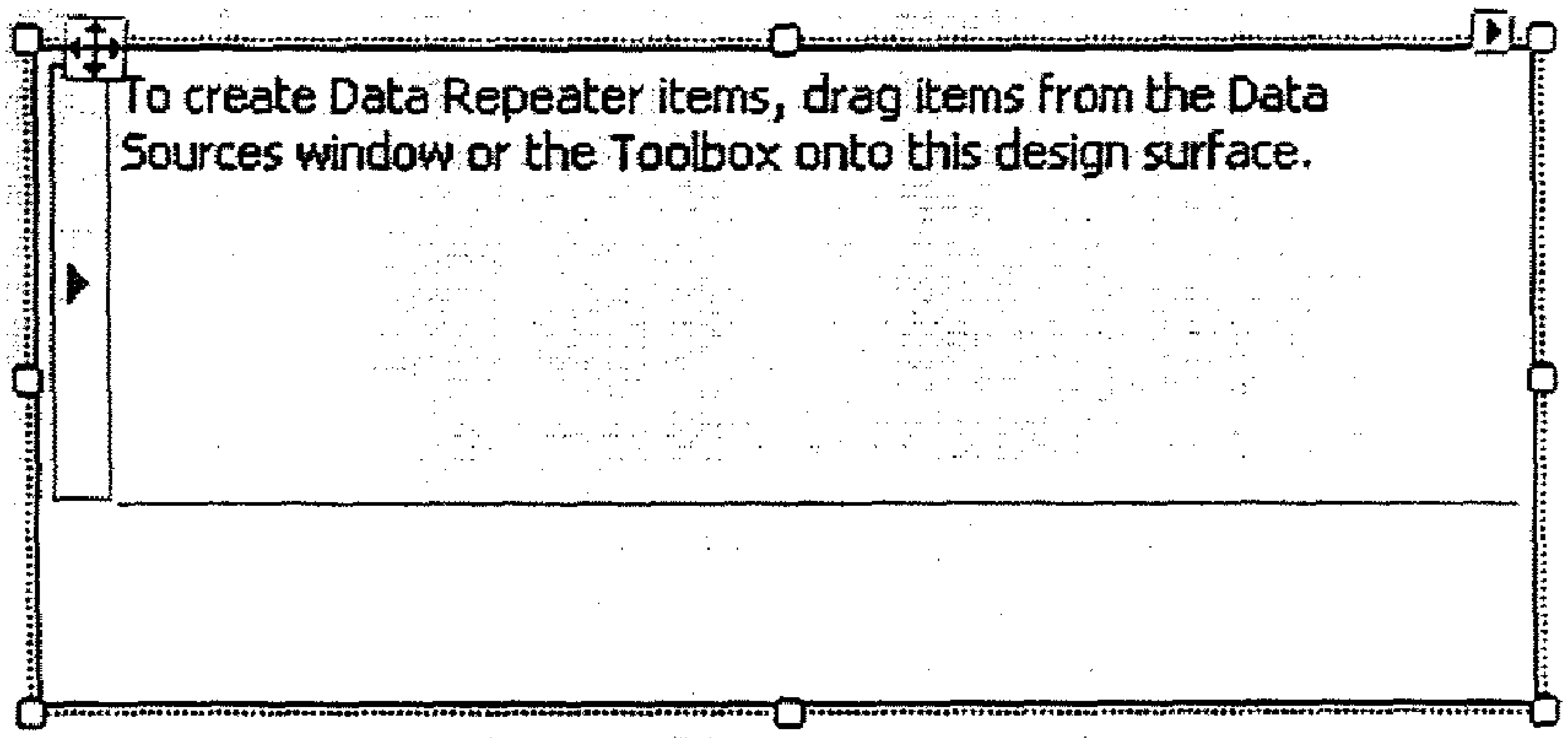
لو تأملت الصورة السابقة، فسيتضح لك أن القائمة التي نتحدث عنها تعرض سجلات جدول المؤلفين، حيث يتم عرض كل مؤلف في ٥ مربعات نصوص مع وجود اللافتات اللازمة التي تشرح وظيفة كل مربع نص.. هذا معناه أننا استخدمنا ١٠ أدوات لتصميم طريقة عرض كل عنصر في القائمة.

لكن.. هل نحن مضطرون إلى تصميم كل العناصر بأنفسنا؟

بالطبع لا، وإلا كان الأمر مستحيلا.. في الحقيقة نحن نصمم عنصرا واحدا فقط في القائمة، ومن ثم يعمل هذا العنصر كقالب Template تنسخ باقي العناصر منه.. لهذا تسمى هذه الأداة بمكرر البيانات DataRepeater، وهي ترث الفئة ContainerControl، لهذا فهي تعمل كأداة حاوية.. هذا هو ما يتيح لنا وضع أدوات أخرى عليها لتصميم عناصر القائمة.

دعنا ونرى كيف نستخدم هذه الأداة لعرض بيانات المؤلفين:

- افتح مشروعا جديدا اسمه Repeater (ستجده ضمن مشاريع هذا الكتاب).
- أضف مصدر بيانات إلى المشروع يحتوي على الجداول Authors و Books و Countries.
- افتح مخطط مجموعة البيانات، وأضف إلى جدول المؤلفين عمودا جديدا اسمه Country، وضع في الخاصية Expression النص: Parent.Name، لتجعل هذا العمود يعرض أسماء الدول التي ينتمي إليها المؤلفون، بدلا من أن نعرض للمستخدم أرقام الدول.
- انتقل إلى النموذج، وافتح صندوق الأدوات، وأسدل عناصر الشريط Visual Basic PowerPacks، واسحب الأداة DataRepeater وأسقطها على النموذج.
- استخدم مقابض الأداة لمنحها الشكل الملائم.. ولو أردت تغيير موضع الأداة فعليك سحبها من علامة الأسهم الموجودة على الركن العلوي الأيسر، وهي لا تظهر إلا إذا ضغطت إطار الأداة بالفأرة، تماما كما تفعل مع أي أداة حاوية.




- كما تلاحظ في الصورة، ينقسم سطح الأداة إلى جزئين:

١- قالب العنصر Item Template:

وهو الجزء العلوي، الذي يوجد سهم أسود على يساره.. ولو ضغطت هذا الجزء فسيتم تحديد إطاره، وسيمكنك تغيير حجمه باستخدام المقابض.. ويمكنك وضع الأدوات من صندوق الأدوات في هذا الجزء، كما يمكنك سحب العناصر من نافذة مصادر البيانات Data Sources وإلقائها عليه مباشرة لإنتاج أدوات مرتبطة بالبيانات.. لاحظ أن مكرر البيانات يشترط وجود أداة واحدة على الأقل مرتبطة بالبيانات، وغير هذا، تستطيع وضع أية أدوات أخرى تريدها، كمربع صورة يعرض صور رمزية، أو زرا ينفذ وظيفة معينة، أو لافتات تشرح وظائف مربعات النصوص.. ويقوم مكرر البيانات بعمل نسخ من هذا القالب، لعرض كل عنصر في مصدر البيانات.

٢- حاوية العرض Viewport:

هذا هو الجزء السفلي الفارغ من الأداة، وعند ضغطه يتم تحديد الأداة كلها.. ولا تستطيع إضافة أية أدوات إلى هذا الجزء، ووظيفته الوحيدة هي تحديد مساحة عرض الأداة على النموذج.. لهذا يمكنك سحب حواف هذا الجزء لضبط المسافات بينها وبين حواف النموذج.. ولا تقلق من صغر مساحة مكرر البيانات، فهو يعرض منزلقا رأسيا إذا احتاج إلى ذلك، ليستطيع المستخدم عرض العناصر غير الظاهرة.. والأفضل أن تستخدم الخاصية Anchor من نافذة الخصائص لتثبيت حواف الأداة بالنسبة لحواف النموذج، بحيث يتم تكبير أو تصغير مكرر البيانات إذا تم تكبير أو تصغير النموذج.

- افتح نافذة مصادر البيانات Data Sources Window، واضغط زر إسدال طريقة العرض المجاور للعنصر Authors واختر Details من القائمة المنسدلة.. واجعل الحقل Country يعرض بياناته في قائمة مركبة ComboBox، ثم اسحب جدول المؤلفين وألقِه على قالب العنصر في مكرر البيانات.. سيؤدي هذا إلى إضافة الأدوات المناسبة إلى مكرر البيانات والنموذج.. احذف مربع النص واللافتة المرتبطتين بالحقل CountryID، ونسق شكل الأدوات كما تريد.
- أضف زرا إلى قالب العنصر، لنستخدمه لعرض كتب المؤلف.

لاحظ أن مكرر البيانات يسبب مشاكل إذا حاولت تصمم عنصر يعرض بيانات مترابطة.. مثلا: لو عرضت كتب المؤلف الحالي في جدول عرض، فسيعرض جدول العرض كتب أول مؤلف في سجلات باقي المؤلفين!!.. وهو نفس ما سيحدث لو حاولت عرض الكتب في قائمة أو قائمة مركبة!!.. بل إنك لو غيرت العنصر المحدد في القائمة المركبة في أحد السجلات، فسيتم تغييره في كل القوائم

المركبة الموجودة في باقي السجلات!!.. بينما لو كانت القائمة المركبة تعرض عناصر عادية (مضافة إلى المجموعة Items دون الارتباط بمصدر بيانات)، فستعمل كل نسخة من القائمة بشكل مستقل وصحيح!.. لهذا علينا أن نكتب بعض الكود لنملأ قائمة الدول.. أضف هذا الكود إلى حدث تحميل النموذج:

```
For Each R As BooksDataSet.CountriesRow In
        BooksDataSet.Countries.Rows
    CountryComboBox.Items.Add(R.Name)
Next
```

لاحظ أنك لو أضفت هذا الكود بعد ملء جدول المؤلفين بالبيانات، فلن تعرض القائمة المركبة أية عناصر، رغم أن العناصر موجودة فيها فعلا!!

السبب في هذا أن ملء جدول المؤلفين بالبيانات يجعل الأدوات المرتبطة به تتلقى البيانات منه، وهذا سيجعل مكرر البيانات يعرض جميع سجلات المؤلفين، وهذا معناه أنه أنشأ نسخا من القائمة المركبة الفارغة من العناصر وعرضها.. لهذا لا يفيدك ملء القائمة الأصلية بعد هذا، فهي ليست مرتبطة فعليا بالنسخ المعروضة للمستخدم.. هي فقط مجرد قالب Template يتم عمل نسخ منه.. لهذا يجب أن تملأ هذا القالب بالبيانات أولا وتضبط خصائص شكله ولون خطه وطريقة عرضه، قبل أن يتم عمل نسخ منه.. هذا معناه أن أفضل مكان لوضع الكود السابق هو بعد جملة ملء جدول الدول وقبل جملة ملء جدول المؤلفين!

وهناك حل آخر لهذه المشكلة، هو استخدام الوسيلتين BeginResetItemTemplate و EndResetItemTemplate كما سنرى لاحقا.

ولا تنسَ أن تستخدم نافذة الخصائص لتجعل قائمة الدول تعمل كقائمة منسدلة وذلك بوضع القيمة DropDownList في الخاصية DropDownStyle.. هذ سيمنع المستخدم من الكتابة في مربع نص القائمة المركبة، حتى لا يكتب اسم دولة خاطئ، وبدلا من هذا سيختار الدولة التي يريدها من القائمة.. لاحظ أن الخاصية Text الخاصة بالقائمة المركبة مرتبطة بالحقل Author.Country بسبب سحبها من نافذة مصادر البيانات.. سنترك هذا كما هو، ولن يحدث خطأ فعندما يوضع في الخاصية Text نص موجود فعلا في القائمة، فإن القائمة تحدد هذا العنصر، وهو ما سيجعل البرنامج يعمل بشكل صحيح.

أما إذا أردت عرض كتب كل مؤلف، فأفضل حل هو استخدام زر يؤدي ضغط إلى عرض نموذج جديد عليه كتب المؤلف الحالي.. وعموما هذه هي الطريقة الأكفأ، فليس من الذكي عرض كما ضخما من البيانات في مكرر البيانات، لأنه ستلتهم مساحة عرض كبيرة وتستهلك مساحة كبيرة في الذاكرة!

لو شغلت البرنامج الآن، فسيعرض بيانات كل مؤلفين في أدوات العرض التي صممناها.. وسنرى ونحن نتعرف على خصائص ووسائل مكرر البيانات كيف نكمل وظائف هذا البرنامج.

## فئة مكرر البيانات DataRepeater Class

هذه الفئة موجودة في النطاق Microsoft.VisualBasic.PowerPacks، وهي ترث الفئة ContainerControl.
وإضافة إلى ما ترثه من الفئة الأم، تمتلك هذه الأداة الخصائص التالية:

### السماح للمستخدم بإضافة عناصر AllowUserToAddItems:

إذا جعلت قيمتها True (وهي القيمة الافتراضية)، فسيتمكن المستخدم من إضافة سجل جديد إلى مكرر البيانات، وذلك بضغط زر الإضافة الموجود على شريط موجه الربط، أو بتحديد أي سجل في مكرر البيانات (بضغط الهامش الأيسر للسجل، حيث سيظهر فيه سهم يدل على أنه محدد)، وضغط CTRL+N من لوحة المفاتيح.
ويعرض السجل الجديد القيم الافتراضية للحقول، وإذا لم تكن للحقل قيمة افتراضية، فستعرض الأدوات قيم أول أو آخر سجل في الجدول.. طبعا هذا غير مرغوب، وعليك التأكد من إفراغ الحقول من هذه القيم، كما سنرى لاحقا.. لاحظ أن السجل الجديد يتم حذفه إذا غادره المستخدم دون أن يكتب فيه أية بيانات.
أما إذا جعلت قيمة هذه الخاصية False، فلن يمكن للمستخدم إضافة سجل جديد بضغط CTRL+N من لوحة المفاتيح، لكن سيظل زر إضافة سجل جديد الموجود على موجه الربط فعالا، وسيكون عليك تعطيله بنفسك.

### السماح للمستخدم بحذف العناصر AllowUserToDeleteItems:

إذا جعلت قيمتها True (وهي القيمة الافتراضية)، فسيتمكن المستخدم من حذف السجل المحدد حاليا في مكرر البيانات، بضغط زر الحذف الموجود على شريط موجه الربط، أو ضغط الزر DELETE من لوحة المفاتيح.

### عدد العناصر ItemCount:

تعيد عدد السجلات المعروضة حاليا في مكرر البيانات.
ويمكنك أن تضع في هذه الخاصية عدد العناصر التي تريد عرضها عند استخدام مكرر البيانات في الوضع الافتراضي Virtual Mode كما سنرى لاحقا.. لكن محاولة وضع أي قيمة في هذه الخاصية في الوضع العادي ستؤدي إلى حدوث خطأ في البرنامج.

## قالب العنصر ItemTemplate:

تعيد كائنا من النوع DataRepeaterItem، يمثل العنصر المستخدم كقالب في مكرر البيانات.. وسنتعرف على الفئة DataRepeaterItem بعد قليل.
ويمكنك استخدام هذه الخاصية لتغيير خصائص عناصر مكرر البيانات.. لاحظ أنك تستطيع فعل هذا في وقت التصميم، وذلك بضغط قالب العنصر بالفأرة لتحديده، ثم ضغط F4 لعرض خصائصه في نافذة الخصائص.. هذا يتيح لك تغيير الخط ولون الخلفية والعديد من الخصائص الأخرى التي تؤثر على المساحة التي تعرض السجلات في مكرر البيانات.. بينما لو ضغطت جزء العرض Viewport فستظهر خصائص مكرر البيانات نفسه في نافذة الخصائص.

## العنصر الحالي CurrentItem:

تعيد كائنا من النوع DataRepeaterItem، يمثل العنصر المحدد حاليا في مكرر البيانات.. ويمكنك أيضا أن تضع في هذه الخاصية، كائن العنصر الذي تريد تحديده.. ولا توجد طريقة لتحديد أكثر من عنصر في نفس الوقت.
لاحظ أن مكرر البيانات لا يمتلك الخاصية الافتراضية Items.. السبب في هذا أن مكرر البيانات هو أداة حاوية، لهذا تستطيع أن تتعامل مع عناصره من خلال الخاصية الموروثة Controls، التي تستطيع أن ترسل إليها رقم العنصر لتعيد إليك الكائن الذي يمثله.. مثال:

```
Dim Itm As DataRepeaterItem = DataRepeater1.Controls(0)
```

والمثال التالي يتيح لك المرور عبر عناصر مكرر البيانات:

```
For Each Itm As DataRepeaterItem In
            DataRepeater1.Controls
    Msgbox(Itm.ItemIndex)
Next
```

ولا تنسَ استخدام جملة التضمين التالية أعلى صفحة الكود قبل تجربة المثال

```
Imports Microsoft.VisualBasic.PowerPacks
```

لكني لا أنصحك باستخدام هذه الطريقة، لأنها ستمر على بعض عناصر مكرر البيانات فقط وبترتيب عشوائي!!.. السبب في هذا أن مكرر البيانات يعرض فقط العناصر الظاهرة للمستخدم على الشاشة، ولا يعرض باقي العناصر إلا إذا حرك المستخدم المنزلق الرأسي.. لذا إذا أردت إجراء أي تغيير على العناصر، فاستخدم الحدث DrawItem لفعل هذا، فهو ينطلق قبل عرض كل عنصر.

**رقم العنصر الحالي CurrentItemIndex:**

تعيد رقم السجل المحدد حاليا في مكرر البيانات.. ويمكنك إرسال رقم أي سجل ليتم تحديده.. والمثال التالي يحدد السجل الثاني في الأداة:

**DataRepeater1.CurrentItemIndex = 1**

**عدد العناصر المعروضة DisplayedItemCount:**

تعيد عدد السجلات الظاهرة للمستخدم حاليا في مكرر البيانات بدون تحريك المنزلق الرأسي.. ولهذه الخاصية معامل منطقي، إذا جعلته True فسيدخل ضمن الحساب السجلات التي تظهر أجزاء منها فقط:

**MsgBox(DataRepeater1.DisplayedItemCount(True))**

أما إن جعلته False، فسيتم حساب عدد السجلات الظاهرة بصورة كاملة:

**MsgBox(DataRepeater1.DisplayedItemCount(False))**

**رقم أول عنصر معروض FirstDisplayedItemIndex:**

تعيد رقم أو سجل ظاهر للمستخدم في مكرر البيانات.

**رأس العنصر مرئي ItemHeaderVisible:**

إذا جعلت قيمتها False، فسيتم إخفاء الهاش الأيسر الذي يعرض رعوس العناصر.. والقيمة الافتراضية True.

**حجم رأس العنصر ItemHeaderSize:**

تقرأ أو تغير عرض الهامش الأيسر الذي يعرض رعوس العناصر.

**لون التحديد SelectionColor:**

تقرأ أو تغير لون الخلفية الذي يعرض في خانة رأس السجل المحدد حاليا.

**طراز المخطط LayoutStyle:**

تقرأ أو تغير طريقة عرض مكرر البيانات، وهي تأخذ إحدى قيمتي المرقم DataRepeaterLayoutStyles التاليتين:

| | |
|---|---|
| Vertical | يتم تكرار العناصر رأسيا (من أعلى إلى أسفل) في شكل صفوف.. هذا هو الوضع الافتراضي. |
| Horizontal | يتم تكرار العناصر أفقيا (من اليسار إلى اليمين) في شكل أعمدة، ويظهر هامش علوي يحمل رعوس هذه الأعمدة.. ويمكنك رؤية هذا في المشروع RepeaterItemColor. |

وتمتلك هذه الأداة الوسائل التالية:

- **إضافة جديد AddNew:**
تضيف سجلا إلى نهاية مكرر البيانات.. وتسبب هذه الوسيلة خطأ إذا كانت للخاصية AllowUserToAddItems القيمة False.

- **حذف من موضع RemoveAt:**
أرسل إلى هذه الوسيلة رقم السجل الذي تريد حذفه من مكرر البيانات.

- **إلغاء التحرير CancelEdit:**
تلغي البيانات التي أدخلها المستخدم في السجل الحالي، وتعيد وضع القيم الأصلية في الأدوات.. هذا مفيد إذا أردت أن تمنح المستخدم القدرة على ضغط الزر Esc من لوحة المفاتيح لإلغاء التغييرات التي أجراها في السجل الحالي.. في هذه الحالة عليك أن تكتب إجراء يستجيب للحدث KeyDown لجميع الأدوات التي تعرض بيانات السجل، وتكب فيه الكود الذي يستدعي هذه الوسيلة إن كان الزر المضغوط هو الزر Esc.. وستجد الكود التالي في الإجراء UserCancelsEdit في المشروع Repaeter، مع ملاحظة أن هذا الإجراء يستجيب للحدث KeyDown لكل مربعات النص والقائمة المركبة أيضا:

```
If e.KeyCode = Keys.Escape Then
  DataRepeater1.CancelEdit( )
End If
```

لاحظ أن التغييرات التي يدخلها المستخدم في أي أداة في السجل الحالي، يتم قبولها بمجرد مغادرة الأداة إلى أية أداة أخرى، في نفس السجل أو في سجل آخر.. هذا معناه أن ضغط الزر ESC سيلغي التغييرات التي حدثت في الأداة الحالية فقط ولن يؤثر على أية أداة أخرى.. ولو غادر المستخدم الأداة التي أجرى فيها التغييرات، ثم عاد إليها وضغط ESC فلن يحدث شيء!

- **تحريك العنصر إلى مجال الرؤية ScrollItemIntoView:**
أرسل إلى هذه الوسيلة رقم السجل الذي تريد تحريك المنزلق إليه ليصير مرئيا للمستخدم.

وتوجد صيغة أخرى، تستقبل معاملا ثانيا، إذا جعلته True، فسيتم تحريك المنزلق بحيث يصير السجل هو أول سجل معروض في مكرر البيانات، مع محاذاة الحافة العلوية للسجل بالحافة العلوية لمكرر البيانات.

- **بدء تغيير قالب العنصر BeginResetItemTemplate:**

كما أشرنا من قبل: أي تغيير تجريه على خصائص الأدوات الداخلة في تكوين قالب العنصر بعد عرض عناصر مكرر البيانات يكون بلا تأثير.. لهذا لو أردت تغيير خصائص أية أداة، أو أردت إجراء تعديلات على القالب نفسه بإضافة أو حذف أدوات من خلال الخاصية ItemTemplate، فعليك أولا أن تستدعي الوسيلة BeginResetItemTemplate لتنبيه مكرر البيانات إلى أن هناك تغييرات ستحدث في طريقة العرض.

- **إنهاء تغيير قالب العنصر EndResetItemTemplate:**

استدع هذه الوسيلة في نهاية الكود الذي يجري تعديلات في قالب العنصر، لإجبار مكرر البيانات على إنعاش العناصر التي يعرضها لتظهر عليها التغييرات التي حدثت.. والكود التالي يغير لون خلفية القائمة إلى الأصفر، ويمكنك تجربته بضغط الزر "تغير لون الخلفية" في المشروع Repeater:

```
DataRepeater1.BeginResetItemTemplate( )
CountryComboBox.BackColor = Color.Yellow
DataRepeater1.EndResetItemTemplate( )
```

جرب وضع علامة التعليق ' أمام السطرين الأول والأخير في الكود السابق واضغط الزر.. ستجد أن لون القائمة لن يتغير.

كما تمتلك هذه الفئة الأحداث التالية:

- **تغير رقم العنصر الحالي CurrentItemIndexChanged:**

ينطلق عندما تتغير قيمة الخاصية CurrentItemIndex من الكود، أو بسبب انتقال المستخدم من سجل إلى آخر في مكرر البيانات.

- **خطأ البيانات DataError:**

ينطلق عند حدوث خطأ في قراءة البيانات من مصدر البيانات، أو في نقل البيانات المحدثة من مكرر البيانات إليه.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterDataErrorEventArgs، وله الخصائص التالية:

| | | |
|---|---|---|
| 🔒 | DataRepeater Item | تعيد عنصر مكرر البيانات DataRepeaterItem الذي يمثل السجل الذي حدث فيه الخطأ. |
| 🔒 | Control | تعيد الأداة التي حدث فيها الخطأ. |
| 🔒 | PropertyName | تعيد اسم خاصية الأداة، التي سببت الخطأ.. بمعنى آخر: تعيد عنصر العرض. |
| 🔒 | Exception | تعيد كائن الاستثناء Exception الذي يحمل معلومات الخطأ. |
| | ThrowException | إذا جعلت قيمة هذه الخاصية True، فسيحدث الخطأ في البرنامج بعد انتهاء هذا الحدث.. والقيمة الافتراضية هي False. |

### أضاف المستخدم عناصر UserAddedItems:

ينطلق بعد أن يضغط المستخدم CTRL+N، وقبل أن يضاف العنصر الجديد إلى مكرر البيانات.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterAddRemoveItemsEventArgs، وله الخاصيتان التاليتان:

| | | |
|---|---|---|
| 🔒 | ItemIndex | تعيد رقم العنصر الجديد. |
| 🔒 | ItemCount | تعيد عدد العناصر التي تمت إضافتها. |

ويعتبر هذا الحدث أنسب مكان لإفراغ خانات السجل من أية قيم غير مرغوبة، فكما ذكرنا سابقا، يعرض مكرر البيانات قيم السجل الأول أو الأخير في السجل الجديد، لهذا يمكنك أن تمحوها، أو تضع بدلا منها القيم الابتدائية المناسبة.. وقد استخدمنا هذا الحدث لفعل هذا في المشروعين Repeater و RepeaterItemColor.

### يجري حذف عناصر DeletingItems:

ينطلق عند حذف سجل من مكرر البيانات، سواء من الكود أو بواسطة المستخدم.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterAddRemoveItemsCancelEventArgs، وهو مماثل لمعامل الحدث السابق، إلا أنه يزيد عنه بامتلاك الخاصية Cancel، وإذا وضعت فيها True يتم إلغاء حذف السجل.. لهذا يعتبر هذا الحدث ملائما لتعرض رسالة للمستخدم ليؤكد رغبته في حذف السجل:

```
If MsgBox("هل تريد حذف هذا السجل فعلا؟",
  MsgBoxStyle.OkCancel) = MsgBoxResult.Cancel Then
      e.Cancel = True
End If
```

لاحظ أنك ضغط زر الحذف الموجود على شريط موجه الربط سيحذف العنصر من مصدر البيانات مباشرة، ولن تظهر رسالة التحذير.. لو أردت تغيير هذا الأداء، فضع في الخاصية DeleteItem الخاصة بموجه الربط القيمة Nothing، واكتب ما يلي في حدث ضغط زر الحذف:

```
Dim I = DataRepeater1.CurrentItem.ItemIndex
DataRepeater1.RemoveAt(I)
```

وستجد هذا الكود في المشروع Repeater.

**المستخدم يحذف عناصر UserDeletingItems:**

مماثل للحدث السابق في كل شيء، ما عدا أنه ينطلق فقط عندما يضغط المستخدم الزر Delete لحذف السجل المحدد في مكرر البيانات، ولا ينطلق بسبب حذف السجل من الكود.

**المستخدم حذف عناصر UserDeletedItems:**

ينطلق بعد أن يحذف المستخدم سجلا من مكرر البيانات، ولا ينطلق بسبب حذف السجل من الكود.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterAddRemoveItemsEventArgs، وقد سبق أن تعرفنا عليه.. هذا معناه أنك لا تستطيع استعادة السجل بعد حذفه، فهذا الحدث لا يملك الخاصية e.Cancel.

**يجري نسخ العنصر ItemCloning:**

ينطلق قبيل عمل نسخة من قالب العنصر.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterItemCloneEventArgs، وهو يمتلك الخصائص التالية:

| | | |
|---|---|---|
| تعيد عنصر مكرر البيانات DataRepeaterItem الذي سيتم نسخه. | Source | 🔒 |
| تقرأ أو تغير عنصر مكرر البيانات DataRepeaterItem الناتج من عملية النسخ.. هذا يتيح لك التحكم في عملية النسخ كما تريد، فالكائن الذي تضعه في هذه الخاصية يكون هو ناتج النسخ. | Target | |

| | | |
|---|---|---|
| | Handled | اجعل قيمتها True، لتمنع مكرر البيانات من أداء عملية النسخ الخاصة به.. في هذه الحالة يجب عليك أن تأخذ نسخة من الكائن الموضح في الخاصية Source، وتجري على هذه النسخة التعديلات التي تريدها، ثم تضعها في الخاصية Target. |

**تم نسخ العنصر ItemCloned:**

ينطلق بعد نسخ عنصر من قالب العناصر.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterItemEventArgs، وهو يمتلك الخاصية DataRepeaterItem التي تعيد العنصر المنسوخ.. لاحظ أن هذا العنصر لم يعرض بعد في مكرر البيانات، لهذا لا تحاول استخدام رقمه في أي عملية، فسيكون صفرا دائما!.. كما أن الأدوات الموجودة على العنصر ما زالت فارغة ولم ترتبط بمصدر البيانات بعد، لهذا لا تحاول قراءة قيمها.. كل ما يمكنك فعله هو تغيير خصائص هذه الأدوات بالطريقة التي تناسبك، كأن تملأ قائمة بمجموعة من العناصر مثلا.

**رسم عنصر DrawItem:**

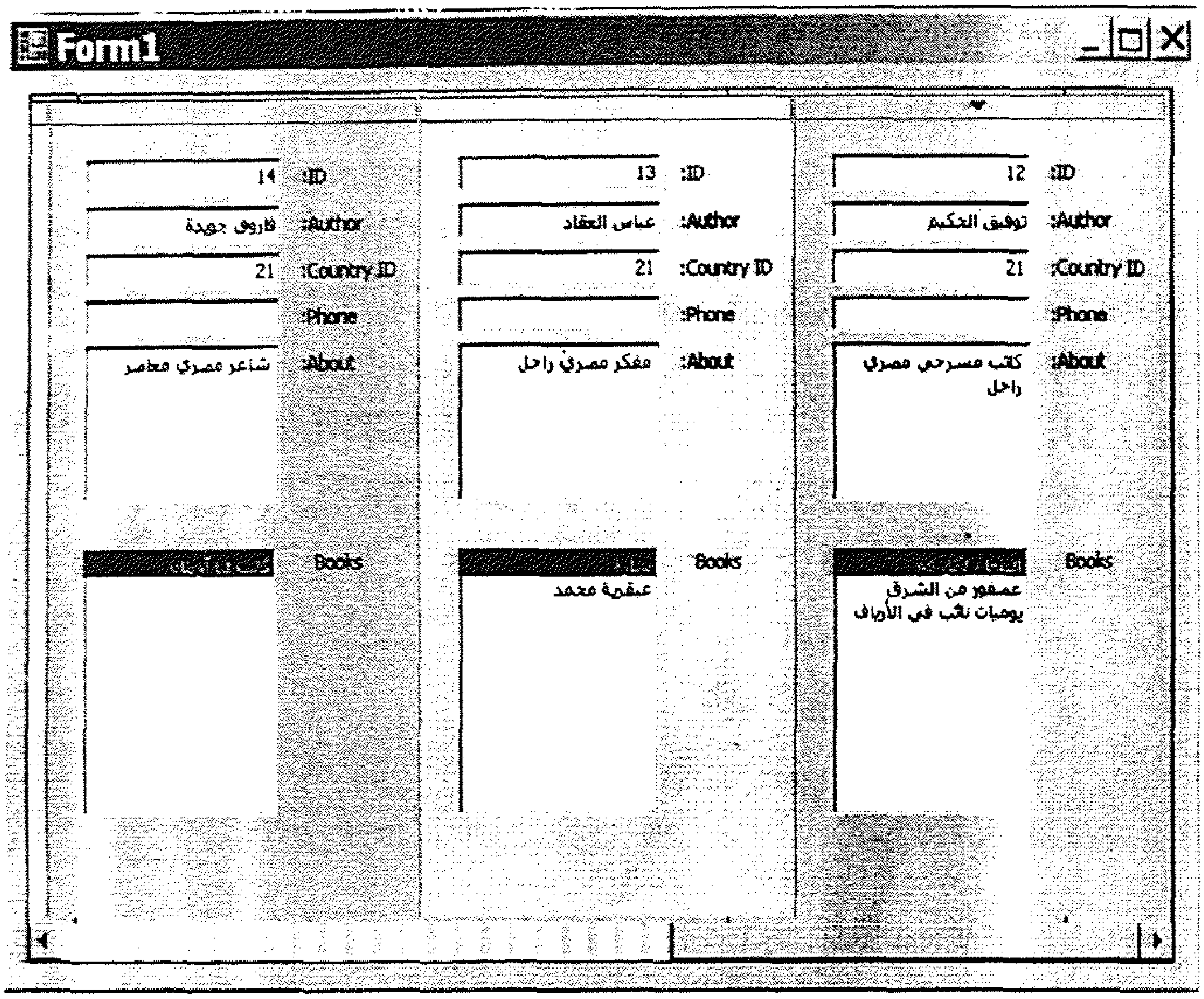
ينطلق عند رسم عنصر في مكرر البيانات.. لاحظ أن رسم العنصر يتكرر مرات عديدة، حيث يعاد رسم العنصر كلما ظهر في مساحة العرض مع حركة المنزلق.

والمعامل الثاني e لهذا الحدث من النوع DataRepeaterItemEventArgs كما في الحدث السابق.

ويعتبر هذا الحدث أفضل حدث يمكنك استخدامه للتحكم في العناصر المعروضة، فهو الحدث الوحيد الذي ينطلق بعد إضافة العنصر إلى مكرر البيانات فعلا وبعد إتمام ربط أدواته بالبيانات.. لهذا استخدمنا هذا الحدث في المشروع RepeaterItemColor لأداء الوظيفتين التاليتين:

١- إذا كان العنصر زوجيا نلونه بالأخضر، وإذا كان فرديا نلونه بالأصفر:

```
If e.DataRepeaterItem.ItemIndex Mod 2 = 0 Then
   e.DataRepeaterItem.BackColor = Color.Green
Else
   e.DataRepeaterItem.BackColor = Color.Yellow
End If
```




٢- نقرأ رقم المؤلف المعروض في مربع النص، ونستخدمه لنحصل على كائن عرض DataView يحتوي على كتبه، ونجعله مصدر البيانات للقائمة لكي تعرض كتب المؤلف الحالي.. لاحظ أن هذه هي الطريقة الصحيحة الوحيدة لربط القائمة بمصدر البيانات، حيث يجب أن نربط كل نسخة من القائمة بمصدر بياناتها على حدة، وقد رأينا من قبل كيف تفشل محاولة ربط القائمة الموضوعة على قالب العنصر بالبيانات.. لكن عيب هذه الطريقة هو أنك مضطر إلى إعادة ربط القائمة بمصدر البيانات في كل مرة يتم فيها رسم العنصر.. ولو جربت الكود التالي، فسيؤدي إلى نتائج خاطئة، وستعرض بعض القوائم كتب مؤلفين آخرين:

```
If BksLst.DataSource Is Nothing Then
    BksLst.DataSource = BooksView
    BksLst.DisplayMember = "Book"
End If
```

السبب في هذا أن الشرط سيكون صحيحا مرة واحدة فقط عند رسم القائمة لأول مرة، لكن بعد هذا كلما تحرك المنزلق وأعيد رسم العنصر، فسيكون الشرط خاطئا، ولن يتم ربط القائمة بمصدر البيانات، مما سيجعلها تعرض نتائج خاطئة.. لست أعرف يقينا سبب هذا، ولكني أخمن أن مصممي مكرر البيانات يحسّنون أداءه بتحريك

القوائم من العناصر التي اختفت مع حركة المنزلق، لعرضها على العناصر التي ظهرت على الشاشة!.. لهذا لو لم تقم بتحديث محتويات كل قائمة بنفسك عند رسم العنصر، فإنها تظل تحتفظ بنتائج تخص سجلات أخرى!

لاحظ أن خطأ سيحدث في الكود الذي كتبناه عند رسم العنصر الجديد، لأنه غير مرتبط بعد بصف في مجموعة البيانات.. لهذا علينا إضافة شرط لإنهاء الكود إذا كان العنصر جديدا.. يمكننا أن نعرف هذا إذا كان مربع النص IDTextBox يحمل رقما سالبا لأنه لم يأخذ رقما تلقائيا بعد:

```
Dim AuthorID As Integer =
    Itm.Controls("IDTextBox").Text
If AuthorID < 0 Then Exit Sub
```

ولإكمال وظيفة البرنامج، سمحنا للمستخدم بالنقر المزدوج بالفأرة على القائمة، واستخدمنا الحدث DoubleClick الخاص بها لنعرض تفاصيل كتب المؤلف الحالي في جدول عرض على نموذج مستقل.. في الحقيقة هذا الكود في منتهى البساطة، فكل ما نفعله فيه هو جعل مصدر بيانات جدول العرض، هو نفسه مصدر بيانات القائمة:

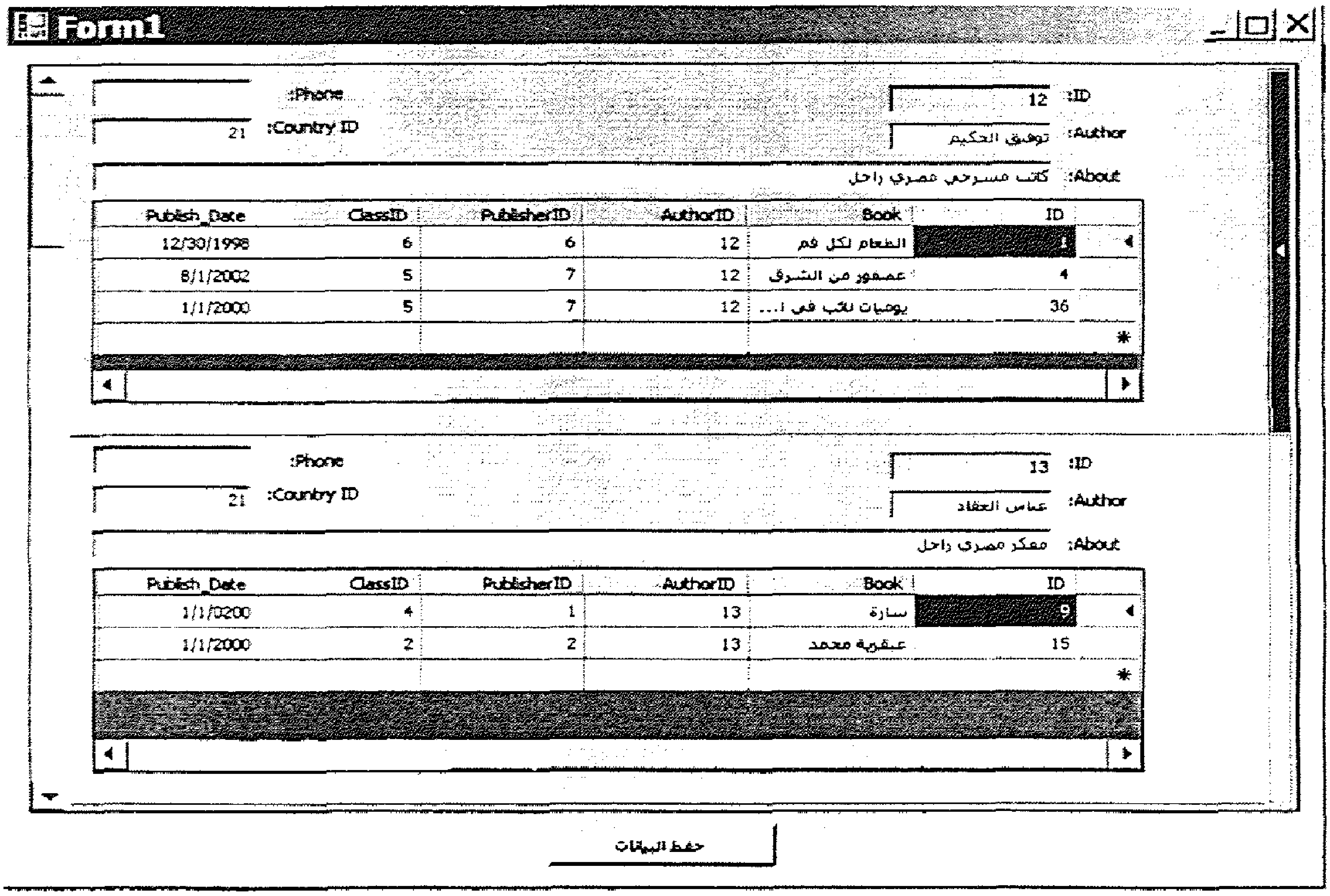
```
FrmBooks.GrdBooks.DataSource = BksLst.DataSource
```

النقطة الوحيدة الهامة هنا، هي أننا لا نستخدم القائمة BooksList الموضوعة على قالب العنصر، وإنما نستخدم نسخة القائمة الخاصة بالعنصر الحالي في مكرر البيانات.. وسنعرف لاحقا كيف نحصل على هذه النسخة.

## استخدام مكرر البيانات في الوضع الافتراضي:

رأينا من قبل كيف نستخدم قائمة العرض ListView وجدول العرض DataGridView في الوضع الافتراضي.. وبالمثل يمكننا استخدام مكرر البيانات في الوضع الافتراضي.. هذا مفيد إذا كنت تولد البيانات بناء على معادلة دون الحاجة إلى مصدر بيانات، أو إذا كان حجم البيانات ضخما، وتريد التدخل في طريقة عرضها لتحسين أداء البرنامج.

ويريك المشروع VirtualRepeater كيف يمكن عرض بيانات المؤلفين في مكرر البيانات بطريقة افتراضية، مع عرض كتب كل مؤلف في جدول عرض في نفس السجل.. في هذه الحالة يحتفظ مكرر البيانات في الذاكرة ببيانات المؤلفين الظاهرين على الشاشة فقط، وكلما تحرك المستخدم بالمنزلق سيطلب منا مكرر البيانات إمداده ببيانات المؤلفين المراد عرضهم.




دعنا نتعرف على الخصائص والوسائل والأحداث التي يمنحها لنا مكرر البيانات للتعامل مع الوضع الافتراضي، لنرى كيف نستخدمها في كتابة هذا المشروع:

### الوضع الافتراضي VirtualMode:

إذا جعلت قيمة هذه الخاصية True، فسيعمل مكرر البيانات في الوضع الافتراضي.. والقيمة الابتدائية لهذه الخاصية هي False.. وقد استخدمنا نافذة الخصائص في المشروع VirtualRepeater لجعل قيمة هذه

الخاصية True.. ونظرا لأن الخاصية ItemCount لا تظهر في نافذة الخصائص، فقد استخدمنا حدث تحميل النموذج لنضع فيها عدد المؤلفين المراد عرضهم:

**DataRepeater1.ItemCount = BooksDataSet.Authors.Count**

لو شغلت المشروع الآن، فسترى عناصر بعدد المؤلفين معروضة في مكرر البيانات.. ورغم أن هذه العناصر ستعرض الأدوات التي وضعتها على قالب العنصر في وقت التصميم، فستكون فارغة، لأن تقنية الربط Binding لا تعمل في الوضع الافتراضي للأسف!!.. لهذا عليك كتابة الكود الذي يعرض البيانات في هذه الأدوات بنفسك، كما سنرى بعد قليل.

### قيمة العنصر مطلوبة ItemValueNeeded:

ينطلق هذا الحدث عندما تحتاج أداة موجودة في أحد السجلات إلى عرض قيمتها.. هذا يشمل اللافتات ومربعات النصوص، لهذا عليك أن تتحقق من الأداة قبل أن تضع فيها القيمة.. ويعتبر هذا الحدث المكان الملائم لعرض البيانات في الأدوات في الوضع الافتراضي.. والمعامل الثاني e لهذه الحدث من النوع DataRepeaterItemValueEventArgs، وله الخصائص التالية:

| | | |
|---|---|---|
| | ItemIndex | تعيد رقم العنصر في مكرر البيانات. |
| | Control | تعيد الأداة التي تحتاج إلى عرض البيانات. |
| | Property Name | تعيد اسم خاصية الأداة التي ستعرض البيانات (عنصر العرض). |
| | Value | ضع في هذه الخاصية القيمة التي تريد عرضها في الأداة.. لاحظ أن هذه الخاصية حساسة جدا لنوع البيانات، لهذا عليك إجراء عمليات التحويل المناسبة قبل وضع القيمة فيها.. مثلا: لو وضعت الرقم ١ في هذه الخاصية لعرضه في مربع النص IDTextBox الذي يعرض رقم المؤلف، فلن يظهر في مربع النص أي شيء!.. بينما لو وضعت النص "١" في هذه الخاصية فسيظهر في مربع النص!.. السبب في هذا أن الخاصية Text تقبل نصوصا لا أعدادا صحيحة، والخاصية e.Value لا تقوم بالتحويل المطلوب!.. لهذا عليك استخدام الوسيلة ToString لتحويل الحقول الرقمية إلى نصوص قبل وضعها في هذه الخاصية. |

وقد استخدمنا هذا الحدث في المشروع VirtualRepeater لعرض القيم في الأدوات.. هذا الكود بسيط للغاية، فهو يستخدم الجملة الشرطية Select ليفحص اسم كل أداة، ويضع فيها القيمة المناسبة.. ولا تحتاج قراءة القيم من جدول المؤلفين إلى كود معقد، فرقم العنصر في مكرر البيانات، هو نفسه رقم السجل في جدول المؤلفين.. سيكون هذا الكود على الصورة التالية:

```
Dim Authors = BooksDataSet.Authors
Select Case e.Control.Name
   Case "IDTextBox"
         e.Value =Authors(e.ItemIndex).ID.ToString
   Case "AuthorTextBox"
         e.Value =Authors(e.ItemIndex).Author
End Select
```

ونظرا لأن بعض الحقول قد تسبب مشاكل إذا كانت فارغة DbNull، لذا عليك استخدام المقطع Try Catch للاحتراز.. وستجد هذا الكود كاملا في المشروع VirtualRepeater.

لاحظ أن جدول العرض لا يطلق الحدث ItemValueNeeded، لهذا عليك استخدام الحدث DrawItem لربط جدول العرض بكتب المؤلف.. كل ما سنفعله، هو الحصول على كائن عرض View Object يحتوي على كتب المؤلف الحالي، ووضعه كمصدر بيانات لجدول العرض:

```
Dim Itm = e.DataRepeaterItem
Dim Authors = BooksDataSet.Authors
' الحصول على كائن عرض الصف الخاص بالمؤلف الحالي
Dim Rv = Authors.DefaultView(Itm.ItemIndex)
Dim Rl = BooksDataSet.Authors.ChildRelations(0)
' الحصول على نسخة جدول العرض الحالية
Dim GrdBooks = CType(Itm.Controls(
      "BooksDataGridView"), DataGridView)
' الحصول على كائن عرض كتب المؤلف الحالي من خلال العلاقة
' واستخدامه كمصدر بيانات لجدول العرض
GrdBooks.DataSource = Rv.CreateChildView(Rl)
```

### عنصر جديد مطلوب NewItemNeeded:

ينطلق هذا الحدث عندما يطلب المستخدم إضافة سجل جديد إلى مكرر البيانات بضغط CTRL+N.. هذا يتيح لك إضافة سجل جديد إلى مصدر البيانات، حتى يمكن حفظ البيانات التي يدخلها المستخدم فيه.. وقد استخدمنا هذا الحدث في المشروع VirtualRepeater لإضافة صف جديد إلى جدول المؤلفين كالتالي:

```
Dim R = BooksDataSet.Authors.NewAuthorsRow
R.Author = " "
R.CountryID = 12
BooksDataSet.Authors.AddAuthorsRow(R)
```

لاحظ أننا وضعنا مسافة في حقل اسم المؤلف، لأن جدول المؤلفين لا يسمح بتركه فارغا، كما وضعنا الرقم ١٢ مبدئيا في حقل رقم الدولة لنفس السبب.. لو لم نفعل هذا، فسيحدث خطأ في البرنامج.. ويمكنك التخلص من المسافة قبل عرضها في مربع النص، باستخدام الوسيلة Trim في الحدث ItemValueNeeded.

### إضافة عنصر ItemsAdded:

ينطلق هذا الحدث بعد إضافة السجل الجديد إلى مكرر البيانات، ليمكنك قراءة رقم العنصر الجديد.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterAddRemoveItemsEventArgs الذي تعرفنا عليه سابقا.. لاحظ أن ترتيب استدعاء الأحداث عند إضافة عنصر جديد كالتالي:

- NewItemNeeded.
- ItemValueNeeded.
- DrawItem.
- ItemsAdded.

### دفع قيمة العنصر ItemValuePushed:

ينطلق هذا الحدث عندما يغير المستخدم قيمة إحدى الأدوات الموضوعة على السجل الحالي، ثم ينتقل منها إلى أداة أخرى.. هذا يتيح لك كتابة الكود المناسب لحفظ قيمة هذه الأداة في مصدر البيانات.. ولا تنسَ فحص القيمة والتأكد من أنها مناسبة قبل محاولة نقلها إلى مصدر البيانات، كي لا يحدث خطأ.. وسيكون من الجيد أن تمنع الخطأ من المنبع، كالتالي:

١- استخدام الخاصية MaxLength لتحديد أقصى طول لمربعات النصوص التي تستقبل نصوصا.. لقد وضعنا الرقم ٣٠ في هذه الخاصية في مربع النص الذي يستقبل اسم المؤلف.
٢- وضع القيمة True في الخاصية ReadOnly لجعل مربع النص الذي يعرض رقم المؤلف ID للقراءة فقط.
٣- كتابة الكود المناسب في الحدث KeyPress في مربعات النص التي تستقبل أرقاما، لمنع كتابة أية حروف.
٤- استخدام أداة التاريخ والوقت DateTimePicker لاستقبال التاريخ بدلا من مربعات النصوص.. كما يمكنك استخدام عمود مخصص لعرض التواريخ في جدول العرض، بالطريقة التي تعلمناها في الفصل الخاص بجدول العرض.
٥- استخدام قائمة منسدلة لعرض أسماء الدول بدلا من السماح للمستخدم بكتابة رقم الدولة.. سأترك لك فعل هذا بنفسك، فقد فعلناه من قبل.
٦- استخدام مربع نص مقنن MaskedTextBox لاستقبال رقم الهاتف بالصيغة الصحيحة (راجع مرجع برمجة الويندوز).

لاحظ أنك لا تحتاج إلى حفظ التغييرات التي تحدث في سجلات جدول العرض، لأنها تحفظ تلقائيا بسبب ربطه بمصدر البيانات.

والمعامل الثاني e لهذا الحدث من النوع DataRepeaterItemValueEventArgs كما في الحدث ItemValueNeeded.

### هل العنصر الحالي قذر IsCurrentItemDirty:

تعيد True إذا كان المستخدم قد أجرى تعديلات على السجل الحالي في مكرر البيانات، دون أن تحفظ بعد في مصدر البيانات.. يحدث هذا إذا غير المستخدم قيمة إحدى الأدوات دون أن يغادرها.

### تم حذف العنصر ItemsRemoved:

ينطلق هذا الحدث بعد حذف عنصر من مكرر البيانات، ليتيح لك حذف العنصر المناظر له من مصدر البيانات.. والمعامل الثاني e لهذا الحدث من النوع DataRepeaterAddRemoveItemsEventArgs، وقد سبق أن تعرفنا عليه.. وقد استخدمنا هذا الحدث في المشروع VirtualRepeater لحذف المؤلف من جدول المؤلفين كالتالي:

```
BooksDataSet.Authors.Rows.RemoveAt(e.ItemIndex)
```

لاحظ أننا لا نحتاج على حذف كتب المؤلف ولن تحدث أية أخطاء لهـذا.. السبب في هذا أننا عرفنا قيد المفتاح الفرعي التالي في حدث تحميل النموذج:

```
Dim Fkc As New ForeignKeyConstraint(
      BooksDataSet.Authors.IDColumn,
      BooksDataSet.Books.AuthorIDColumn)
Fkc.UpdateRule = Rule.Cascade
Fkc.AcceptRejectRule = Rule.Cascade
Fkc.DeleteRule = Rule.Cascade
BooksDataSet.Books.Constraints.Add(Fkc)
```

كما ترى، فقد عرفنا قاعدة الحذف المتتالي، لحذف كتـب المؤلـف تلقائيـا بمجرد حذف المؤلف نفسه، وهذا يمنع حدوث أية أخطاء، ويوفر علينا كتابة كود الحذف.. كما عرفنا قاعدة التحديث المتتالي أيضا لمنع أية مشكلة عنـد حفظ بيانات المؤلف الجديد وتغير رقمه التلقائي.. والحقيقـة أن المـستخدم يجب ألا يدخل الكتب قبل حفظ المستخدم، وإلا فقد يخسرها بسبب عدم قبول قيمة الحقل AuthorID بعد تغير الرقم التلقائي للمؤلف.

## فئة عنصر مكرر البيانات DataRepeaterItem Class

هذه الفئة ترث فئة اللوحة Panel، لهذا تستطيع احتواء أدوات أخرى، وهي تعمل كعنصر موضوع على مكرر البيانات، سواء كان العنصر المعروض في وقت التصميم (القالب)، أو العناصر المنسوخة منه في وقت التشغيل.
وإضافة إلى ما ترثه من خصائص الأدوات التقليدية وخصائص الأداة الحاوية وخصائص اللوحة، تمتلك هذه الفئة الخصائص التالية:

### هل هو العنصر الحالي IsCurrent:

تعيد True إذا كان هذا العنصر هو العنصر الحالي (المحدد) في مكرر البيانات.

### هل هو قذر IsDirty:

تعيد True إذا كان المستخدم قد غير بعض بيانات العنصر ولم تحفظ التغييرات بعد في مصدر البيانات.

### رقم العنصر ItemIndex:

تعيد رقم العنصر في مكرر البيانات.

وأهم ما يعنينا هنا، هو كيفية التعامل مع الأدوات الموضوعة على العنصر.. كما ذكرنا من قبل، فإن العنصر هو لوحة Panel، وهذا معناه أنه أداة حاوية، لهذا يمكنك استخدام الخاصية Controls للتعامل مع الأدوات الموضوعة عليه سواء بأرقامها أو بأسمائها.. ويكون التعامل مع الأدوات بأرقامها مناسبا إذا أردت المرور عبر كل الأدوات، بينما يكون التعامل مع الأدوات بأسمائها أكثر ملائمة للكود الذي يقرأ قيم الأدوات أو يغيرها، لأنه يجعل الكود أكثر وضوحا وسهولة.. لاحظ أن الاسم الذي تمنحه للأداة في قالب العنصر في وقت التصميم، هو نفسه الاسم الذي ستستخدمه للتعامل مع نسخة الأداة الموضوعة على أي عنصر.. السبب في هذا أن كل عنصر هو نسخة طبق الأصل من القالب، وهذا معناه أن كل أداة موضوعة عليه تأخذ نفس خصائص نسختها الأصلية الموضوعة على القالب بما في ذلك الاسم.. وعليك ألا ترتبك بين الاسم الموضوع في الخاصية Name، واسم المتغير الذي يشير إلى الأداة.. مثلا: الكود التالي يغير نص مربع النص الأصلي الموضوع على القالب:

```
AuthorTextBox.Text = "Test"
```

ولن يؤثر هذا الكود على نسخ مربع النص الموضوعة على الأدوات، إلا إذا استخدمته بين الوسيلتين BeginResetItemTemplate و EndResetItemTemplate كما أوضحنا من قبل.

أما إذا أردت تغيير اسم المؤلف الحالي فقط، فيمكنك استخدام الكود التالي:

```
Dim Itm = DataRepeater1.CurrentItem
Dim AutherTxtBx = CType(
    Itm.Controls("AuthorTextBox"), TextBox)
AutherTxtBx.Text = "Test"
```

وقد استخدمنا الكود التالي في المشروع RepeaterItemColor لعرض كتب المؤلف في القائمة:

```
Dim BksLst As ListBox = Itm.Controls("BooksList")
BksLst.DataSource = BooksView
BksLst.DisplayMember = "Book"
```

كما استخدمنا الكود التالي في حدث ضغط زر عرض كتب المؤلف الموضوع على قالب العنصر:

```
Dim Itm = DataRepeater1.CurrentItem
' الحصول على كائن عرض صف المؤلف الحالي
Dim Rv = BooksDataSet.Authors.DefaultView(Itm.ItemIndex)
' الحصول على كائن عرض يحتوي على كتب هذا المؤلف
Dim Rl = BooksDataSet.Authors.ChildRelations(0)
Dim BooksView = Rv.CreateChildView(Rl)
' عرض الكتب في جدول العرض
FrmBooks.GrdBooks.DataSource = BooksView
FrmBooks.Text = "كتب " &
    Itm.Controls("AuthorTextBox").Text
FrmBooks.ShowDialog( )
```

لاحظ أن كل نسخ الزر تستجيب أيضا لهذا الحدث.. ألم أقل لك إن نسخة الأداة مماثلة للأداة الأصلية في كل شيء؟.. هذا يشمل الإجراءات المستجيبة لأحداث الأداة، لهذا تستطيع برمجة أحداث الأدوات الموضوعة على قالب العنصر مباشرة، وستكون بذلك قد برمجت أحداث كل النسخ المنسوخة من هذه الأداة.

## خاتمة:

إلى هنا، نكون قد وصلنا إلى نهاية رحلتنا مع مكرر البيانات، وإلى نهايـــة هــذا الكتاب أيضا.. لكن رحلتنا مع قواعد البيانات لم تنته بعد، فما زالت أمامنا بعض المواضيع المتقدمة التي سنتعرف عليها في الكتاب التالي بإذن الله، مثل:

- التعاملات Transactions.
- استخدام الاستعلام المتكامل مع قواعد البيانات LinQ To SQL.
- إطار عمل الكينونات Entity Framework.
- عناصر قواعد البيانات المدارة، مثـــل الإجـــراءات المخزنـــة المـــدارة Managed Stored Procedures.
- التقارير الجاهزة Crystal Reports.

فإلى الملتقي في كتاب "المواضيع المتقدمة في برمجة قواعد البيانات" بإذن الله.

## الفهرس


- مقدمة الناشر ٣
- مقدمة المؤلف ٥
- لمن هذا الكتاب ٦

-١-

### تركيب قواعد البيانات Database Structure

قاعدة البيانات ٩
الجداول Tables ١٠
قواعد البيانات المترابطة Relational Databases ١٢
العلاقات Relations ١٥
القيود Constrains ١٩
العروض Views ١٩
الفهارس Indices ٢٠

-٢-

### قواعد بيانات Access

إنشاء الجداول بتطبيق Access ٢٣
إنشاء علاقات بين الجداول ٢٨
إنشاء قيد Constraint ٣١
ضغط قاعدة البيانات Database Compacting ٣٢
متصفح الخوادم Server Explorer ٣٣
الاتصال بقواعد بيانات Access ٣٤

-٣-

### قواعد بيانات SQL Server

إعداد Management Studio Express ٣٨
إنشاء قاعدة بيانات سيكويل سيرفر ٤٢
أنواع البيانات في سيكويل سيرفر ٤٩
إنشاء العلاقات في سيكويل سيرفر ٥٥
التحكم في خادم سيكويل ٦٠
توصيل وفصل قاعدة بيانات ٦١
إنشاء قاعدة بيانات سيكويل سيرفر في دوت نت ٦٢

الاتصال بقواعد بيانات SQL Server ٦٥
إنشاء الفهارس Indices ٦٧
إنشاء القيود Constraints ٦٩

-٤-

## لغة الاستعلام المركّبة Structured Query Language (SQL)

باني الاستعلام Query Builder ٧٢
استعلامات التحديد Selection Queries ٧٦
جملة التحديد SELECT Statement ٧٦
دوال التجميع Aggregate functions ٨٩
عمليات الربط SQL Joins ٩٥
استعلامات الأداء Action Queries ٩٨
حذف الصفوف باستخدام الأمر DELETE ٩٩
إدراج سجلات جديدة باستخدام الأمر INSERT ١٠٠
تحرير السجلات الموجودة باستخدام الأمر UPDATE ١٠٢
التحرير المتتابع للسجلات، باستخدام الأمر UPDATE .WRITE ١٠٣
لغة تعريف البيانات Data Definition Language (DDL) ١٠٦
الإجراءات المخزّنة Stored Procedures ١٠٧
إنشاء الإجراءات المخزنة في قواعد بيانات Access ١١٠
دوال SQL التي يعرفها المستخدم User Defined SQL Functions ١١١
أنواع الجداول التي يعرفها المستخدم User-Defined Table Types ١١٢

-٥-

## تقنية ADO.NET

الخادم Server والعميل Client ١١٥
تقنية ADO.NET ١١٧
لغة XML ١٢٠
مزودات قواعد البيانات Database Providers ١٢٠

-٦-

## أنواع بيانات سيكويل المدارة Managed SQL Data Types

سجل القيمة المنطقية SqlBoolean Structure ١٢٥
سجل الوحدة الثنائية SqlByte Structure ١٢٩

سجل الأعداد العشرية SqlDecimal Structure ١٣٠
فئة الحروف SqlChars Class ١٣١
سجل النص SqlString Structure ١٣٤
سجل البيانات الثنائية SqlBinary Structure ١٣٨
فئة الوحدات الثنائية SqlBytes Class ١٤٠
فئة "XML" SqlXml Class ١٤١
حفظ الملفات خارج قاعدة البيانات ١٤٢
فئة مجرى بيانات سيكويل SqlFileStream Class ١٥٠

-٧-

كائن الاتصال Connection Object

نص الاتصال Connection String ١٥٣
فئة باني نص الاتصال DbConnectionStringBuilder Class ١٥٥
فئة باني نص اتصال سيكيول SqlConnectionStringBuilder Class ١٥٨
حفظ نص الاتصال في إعدادات البرنامج Settings ١٦٨
فئة مقطع نصوص الاتصال ConnectionStringsSection Class ١٧١
فئة إعدادات نص الاتصال ConnectionStringSettings Class ١٧٢
واجهة الاتصال بقواعد البيانات IDbConnection Interface ١٧٤
فئة الاتصال DbConnection Class ١٧٧
فئة اتصال سيكيول SqlConnection Class ١٧٩
فئة خطأ سيكيول SqlError Class ١٨٤

-٨-

كائن الأمر Command Object

واجهة أمر قاعدة البيانات IDbCommand Interface ١٨٥
فئة أمر قاعدة البيانات DbCommand Class ١٩٢
فئة أمر سيكويل SqlCommand Class ١٩٢
تمرير القيم إلى جمل الاستعلام ١٩٥
دس الاستعلامات SQL Injection ١٩٦
المعاملات Parameters ١٩٩
فئة مجموعة معاملات قاعدة البيانات DbParameterCollection ٢٠١
فئة مجموعة معاملات سيكويل SqlParameterCollection Class ٢٠١

- واجهة معامل البيانات IDataParameter Interface ٢٠٣
- واجهة معامل بيانات قاعدة البيانات IDbDataParameter Interface ٢٠٧
- فئة معامل قاعدة البيانات DbParameter Class ٢٠٨
- فئة معامل سيكويل SqlParameter Class ٢٠٩


## -٩-
## قارئ البيانات DataReader

- واجهة سجل البيانات IDataRecord Interface ٢١٦
- فئة سجل البيانات DbDataRecord Class ٢١٩
- واجهة قارئ البيانات IDataReader Interface ٢١٩
- فئة قارئ البيانات DbDataReader Class ٢٢٢
- فئة قارئ بيانات سيكويل SqlDataReader Class ٢٢٤


## -١٠-
## موصل البيانات DataAdapter

- واجهة موصل البيانات IDataAdapter Interface ٢٢٨
- واجهة موصل بيانات قاعدة البيانات IDbDataAdapter Interface ٢٣٤
- فئة موصل البيانات DataAdapter Class ٢٣٧
- فئة موصل بيانات قاعدة البيانات DbDataAdapter Class ٢٤٤
- فئة موصل بيانات سيكويل SqlDataAdapter Class ٢٤٧
- التصارع على تحديث البيانات ٢٥١
- معالج إعداد موصل البيانات Data Adapter Configuration Wizard ٢٦١
- فئة باني أوامر قاعدة البيانات DbCommandBuilder Class ٢٦٨
- فئة باني أوامر سيكويل SqlCommandBuilder Class ٢٧٣
- واجهة مجموعة خرائط الجداول ITableMappingCollection ٢٧٦
- فئة مجموعة خرائط الجداول DataTableMappingCollection Class ٢٧٨
- واجهة خريطة الجدول ITableMapping Interface ٢٧٩
- فئة خريطة الجدول DataTableMapping Class ٢٨٠
- واجهة مجموعة خرائط العمود IColumnMappingCollection ٢٨٢
- فئة مجموعة خرائط العمود DataColumnMappingCollection ٢٨٤
- واجهة خريطة العمود IColumnMapping Interface ٢٨٥
- فئة خريطة العمود DataColumnMapping Class ٢٨٦

-١١-

## مصانع المزودات Provider Factories


فئة مصانع المزودات DbProviderFactories Class ٢٨٨

فئة مصنع المزود DbProviderFactory Class ٢٨٩

الطبقات المتعددة N-Tiers ٢٩٣

فئة عداد مصادر البيانات DbDataSourceEnumerator Class ٢٩٤

فئة عداد مصادر بيانات سيكويل سيرفر SqlDataSourceEnumerator ٢٩٦


-١٢-

## مجموعة البيانات DataSet


فئة مجموعة البيانات DataSet Class ٢٩٩

المعالج السحري لإنشاء مجموعة البيانات Generate DataSet Wezard ٣١٤

إنشاء مجموعات بيانات خاصّة Custom DataSet ٣٢٠

حفظ بيانات الشجرة في مجموعة البيانات ٣٢٥

فئة موصل الجدول TableAdapter Class ٣٢٩

فئة مدير موصلات الجداول TableAdapterManager ٣٤٣


-١٣-

## الجداول والعلاقات والقيود


فئة أساس مجموعة البيانات الداخلية InternalDataCollectionBase Class ٣٥١

فئة مجموعة الجداول DataTableCollection Class ٣٥٢

فئة جدول البيانات DataTable Class ٣٥٥

فئة مجموعة الصفوف DataRowCollection Class ٣٧٢

فئة صفّ البيانات DataRow Class ٣٧٤

فئة مجموعة الأعمدة DataColumnCollection Class ٣٨٢

فئة عمود البيانات DataColumn Class ٣٨٥

فئة قارئ جدول البيانات DataTableReader Class ٣٩٣

فئة مجموعة العلاقات DataRelationCollection Class ٣٩٤

فئة العلاقة DataRelation Class ٣٩٦

فئة مجموعة القيود ConstraintCollection Class ٣٩٩

فئة القيد Constraint Class ٤٠٢

فئة قيد التفرّد UniqueConstraint Class ٤٠٣
فئة قيد المفتاح الثانوي ForeignKeyConstraint Class ٤٠٤

-١٤-

**عروض البيانات Data Views**

واجهة قائمة الربط IBindingList Interface ٤٠٨
واجهة القائمة محددة النوع ITypedList Interface ٤١٢
فئة مدير العرض DataViewManager Class ٤١٣
فئة إعدادات العرض DataViewSetting Class ٤١٥
واجهة ربط قائمة العرض IBindingListView Interface ٤١٩
فئة واصف ترتيب القائمة ListSortDescription Class ٤٢٠
فئة عرض البيانات DataView Class ٤٢١
واجهة الكائن القابل للتحرير IEditableObject Interface ٤٢٧
واجهة التنبيه بتغير خاصية INotifyPropertyChanged Interface ٤٢٨
فئة عرض صف البيانات DataRowView Class ٤٢٨

-١٥-

**ربط البيانات Data Binding**

واجهة المكون القابل للارتباط IBindableComponent Interfac ٤٣٢
فئة مجموعة الارتباطات BindingsCollection Class ٤٣٤
فئة مجموعة ارتباطات الأداة ControlBindingsCollection Class ٤٣٥
فئة الارتباط Binding Class ٤٤٠
سجل معلومات عنصر الربط BindingMemberInfo Structure ٤٤٦
فئة محتوى الربط BindingContext Class ٤٤٧
فئة أساس مدير الربط BindingManagerBase Class ٤٤٩
فئة مدير الخاصية PropertyManager Class ٤٥٤
فئة مدير التسلسل CurrencyManager Class ٤٥٤
ربط الأدوات في وقت التصميم ٤٥٥
ربط مربعات القوائم Binding List Boxs ٤٥٨
معالج تهيئة مصادر البيانات Data Source Configuration Wizard ٤٦٣
متصفح مصادر البيانات ٤٦٧
واجهة مزود مدير التسلسل ICurrencyManagerProvider Interface ٤٧٤

- واجهة إلغاء إضافة الجديد ICancelAddNew Interface ٤٧٤
- واجهة إطلاق أحداث التغير IRaiseItemChangedEvents Interface ٤٧٥
- فئة قائمة الربط عامة النوع BindingList(Of T) Class ٤٧٥
- واجهة مصدر القائمة IListSource Interface ٤٧٦
- فئة مصدر الربط BindingSource Class ٤٧٧
- فئة مساعد ربط القوائم ListBindingHelper Class ٤٨٤
- فئة موجه الربط BindingNavigator Class ٤٨٥


-١٦-

**جدول عرض البيانات DataGridView**


- فئة عنصر جدول العرض DataGridViewElement Class ٤٩٤
- فئة نطاق جدول العرض DataGridViewBand Class ٤٩٥
- فئة أساس المجموعة BaseCollection Class ٤٩٨
- فئة مجموعة أعمدة الجدول DataGridViewColumnCollection ٤٩٩
- فئة عمود جدول العرض DataGridViewColumn Class ٥٠٣
- فئة عمود مربعات النصوص DataGridViewTextBoxColumn ٥١٠
- فئة عمود الأزرار DataGridViewButtonColumn Class ٥١١
- فئة عمود مربعـــات الاختيـــار DataGridViewCheckBoxColumn Class ٥١٢
- فئة عمود الصور DataGridViewImageColumn Class ٥١٤
- فئة عمود الوصلات DataGridViewLinkColumn Class ٥١٦
- فئة عمود القوائم المركبة DataGridViewComboBoxColumn ٥١٨
- فئة مجموعة صفوف جدول العرض DataGridViewRowCollection ٥٢٢
- فئة صف جدول العرض DataGridViewRow Class ٥٢٧
- فئة خانة جدول العرض DataGridViewCell Class ٥٣٣
- فئة خانة مربع النص DataGridViewTextBoxCell Class ٥٤٣
- فئة خانة الزر DataGridViewButtonCell Class ٥٤٣
- واجهة خانة التحرير IDataGridViewEditingCell Interface ٥٤٤
- فئة خانة مربع الاختيار DataGridViewCheckBoxCell Class ٥٤٥
- فئة خانة الصور DataGridViewImageCell Class ٥٤٦
- فئة خانة الوصلة DataGridViewLinkCell Class ٥٤٧
- فئة خانة القائمة المركبة DataGridViewComboBoxCell Class ٥٤٨

- واجهة أداة التحرير IDataGridViewEditingControl Interface ٥٤٩
- فئة أداة تحرير مربع النص DataGridViewTextBoxEditingControl ٥٥١
- فئة أداة تحرير القائمة DataGridViewComboBoxEditingControl ٥٥١
- فئة الخانة الرئيسية DataGridViewHeaderCell Class ٥٥٢
- فئة خانة رأس العمود DataGridViewColumnHeaderCell ٥٥٢
- فئة الخانة العلوية اليسرى DataGridViewTopLeftHeaderCell ٥٥٣
- فئة خانة رأس الصف DataGridViewRowHeaderCell ٥٥٣
- فئة طراز خانة جدول العرض DataGridViewCellStyle Class ٥٥٤
- فئة طراز الحافة المتطور DataGridViewAdvancedBorderStyle ٥٥٧
- فئة جدول عرض البيانات DataGridView Class ٥٥٨

التعامل مع أعمدة جدول العرض ٥٥٨
التعامل مع صفوف جدول عرض البيانات ٥٧٠
التعامل مع خانات جدول عرض البيانات ٥٨٨
التعامل مع جدول العرض ٦٠٤
التعامل مع جدول العرض في الوضع الافتراضي VirtualMode ٦١٤
تحسين أداء جدول العرض ٦٢٠
الصفوف المشتركة Shared Rows ٦٢٢
تقسيم جدول العرض إلى صفحات Paging ٦٢٥


## -١٧-
## شبكة البيانات DataGrid


- واجهة خدمة التحرير IdataGridEditingService Interface ٦٣٨
- فئة طراز شبكة البيانات DataGridTableStyle Class ٦٣٩
- واجهة التنبيه بتحرير عمود شبكة البيانات IDataGridColumnStyleEditingNotificationService Interface ٤٤٢
- فئة طراز العمود DataGridColumnStyle ٤٤٢
- فئة عمود النصوص DataGridTextBoxColumn Class ٦٤٤
- فئة العمود المنطقي DataGridBoolColumn Class ٦٤٥

سجل خانة الشبكة DataGridCell Structure ٦٤٦
فئة شبكة البيانات DataGrid Class ٦٤٧

## -١٨-
## مُكرِّر البيانات Data Repeater

فئة مكرر البيانات DataRepeater Class ٦٦٣
استخدام مكرر البيانات في الوضع الافتراضي ٦٧٣
فئة عنصر مكرر البيانات DataRepeaterItem Class ٦٧٩

**صدر للكاتب في مجال البرمجة:**

- فيجيوال بيزيك وسي شارب: طريقك المختصر للانتقال من إحدى اللغتين إلى الأخرى.
- من الصفر إلى الاحتراف: فيجيوال بيزيك دوت نت ٢٠١٠.
- من الصفر إلى الاحتراف: برمجة إطار العمل (VB.NET 2008).
- من الصفر إلى الاحتراف: برمجة نماذج الويندوز (VB.NET 2008).
- من الصفر إلى الاحتراف: برمجة قواعد البيانات (VB.NET 2010).
- من الصفر إلى الاحتراف: سي شارب ٢٠٠٨.
- من الصفر إلى الاحتراف: برمجة إطار العمل (C# 2008).
- من الصفر إلى الاحتراف: برمجة نماذج الويندوز (C# 2008).
- من الصفر إلى الاحتراف: برمجة قواعد البيانات (C# 2010).

**للمراسلة:**


البريد الالكتروني: **msvbnet@hotmail.com**

مدونة رفاق الخطر: **http://refaq.maktoobblog.com**

مدونة مشروع الترجمة الجامعية **http://uni-trans.blogspot.com**

# قواعد البيانات

احصل
على القرص الضوئي
الخاص بأمثلة الكتاب
مجانا

## في فيجيوال بيزيك دوت نت ٢٠١٠

يفترض هذا الكتاب أنك لا تملك أية معرفة مسبقة بقواعد البيانات والبرامج التي تنشئها بها، لهذا فهو يبدأ معك من الصفر، ليعرفك على المفاهيم الأساسية اللازمة للتعامل مع قواعد البيانات، وكيف تنشئها في Access و SQL Server 2008، وكيف تكتب الاستعلامات التي تحصل بها على البيانات باستخدام لغة SQL.. بعد هذا يعلمك الكتاب كيف تتعامل مع قواعد البيانات من داخل مشاريع فيجيوال بيزيك دوت نت باستخدام تقنية ADO.NET.

ويغطي هذا الكتاب بالتفصيل حوالي ١٣٥ واجهة وفئة وسجلا من مكتبة إطار العمل، كما يشرح بالتفصيل أكثر من ٥٠ مشروعا متنوعا تغطي محتوياته، لتتعلم من خلالها:

- كيف تنقل البيانات بين نوعين مختلفين من قواعد البيانات.
- كيف تحفظ البيانات الثنائية Binary Data في ملفات مستقلة على الخادم خارج قاعدة البيانات في SQL Server 2008.
- كيف تعرّف المعاملات Parameters والمعاملات الجدولية Table-Valued Parameters، وكيف تستخدمها لتمرير البيانات إلى الإجراءات المخزنة في SQL Server 2008.
- كيف تحمي قاعدة البيانات من القراصنة الذين يحاولون دس الاستعلامات SQL Injection من خلال مشروعك.
- كيف تقرأ البيانات الثنائية والنصية الضخمة تتابعيا Sequentially على صورة أجزاء من قواعد بيانات SQL Server 2008.
- كيف تنشئ الإجراءات المخزنة في Access.
- كيف تحفظ البيانات في ملف XML وكيف تستعيدها منه مرة أخرى.
- كيف تتعامل مع علاقة واحد بمتعدد One-To-Many Relation، وعلاقة متعدد بمتعدد Many-To-Many Relation، والعلاقة الذاتية Self Relation.
- كيف تستخدم مصانع المزودات Provider Factories لكتابة فئات عامة قادرة على التعامل مع أي نوع من قواعد البيانات، مما يختصر الكود الذي تكتبه، ويمهد لك الطريق لإنشاء مشاريع متعددة الطبقات N-Tier Applications.
- كيف تحل مشاكل تصارع أكثر من مستخدم على حفظ البيانات في نفس اللحظة باستخدام التطابق المتفائل Optimistic Concurrency.
- كيف تنشئ أنواعا جديدة من أعمدة جدول العرض، تعرض خاناتها أداة اختيار التاريخ أو شجرة منسدلة أو أي نوع آخر تريده من الأدوات.
- كيف تجعل جدول العرض يعمل في الوضع الافتراضي Virtual Mode وكيف تضيف إليه تقنية تقسيم السجلات على صفحات عرض مختلفة Paging.
- كيف تحسن أداء برنامجك بتوفير أكبر قدر من الذاكرة، وكيف تحافظ على كفاءة خادم البيانات، بتقليل عدد الاتصالات، ووقت كل اتصال بقدر الإمكان.

لكل هذا يعتبر الكتاب مرجعا مفصلا مبوبا، يمكن لقارئه الرجوع إليه عند البحث عن تفاصيل أي فئة أو خاصية أو وسيلة أو حدث، في نفس الوقت الذي يعتبر صالحا للقراءة ككتاب تعليمي عملي مرتب من الأسهل إلى الأصعب، ينقل إلى المبرمج في صفحات معدودات خبرة سنوات في برمجة تطبيقات قواعد البيانات، ويرشده إلى كيفية حل المشكلات غير المتوقعة التي تواجهه في هذا المجال.

باختصار: هذا هو الكتاب الذي تبحث عنه.

21389578
Bibliotheca Alexandrina
07437778

الناشر

**مكتبة دار المعرفة**

القاهرة - العباسية : ٤ ش السريات

أمام هندسة عين شمس ت :٢٦٨٤٤٠٤٣

كتب ومراجع علمية . طباعة . نشر . استيراد

E-mail : dar_elmaarefa@yahoo.com